ABC INFORMATIQUE

N° 86

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE

Intelligence artificielle

Quand la Chine...

Traitement des fichiers en PASCAL

La puce son du C 64

EDITIONS ATLAS

M 6062-86-12,50 F 90 FB - 3,80 FS - $ 1.95

Page manquante
(publicité et colophon)

C'est la tête

Le domaine de l'intelligence artificielle représente sans doute l'aspect le plus passionnant de l'informatique. Voici des techniques de base qui peuvent s'appliquer à votre micro.

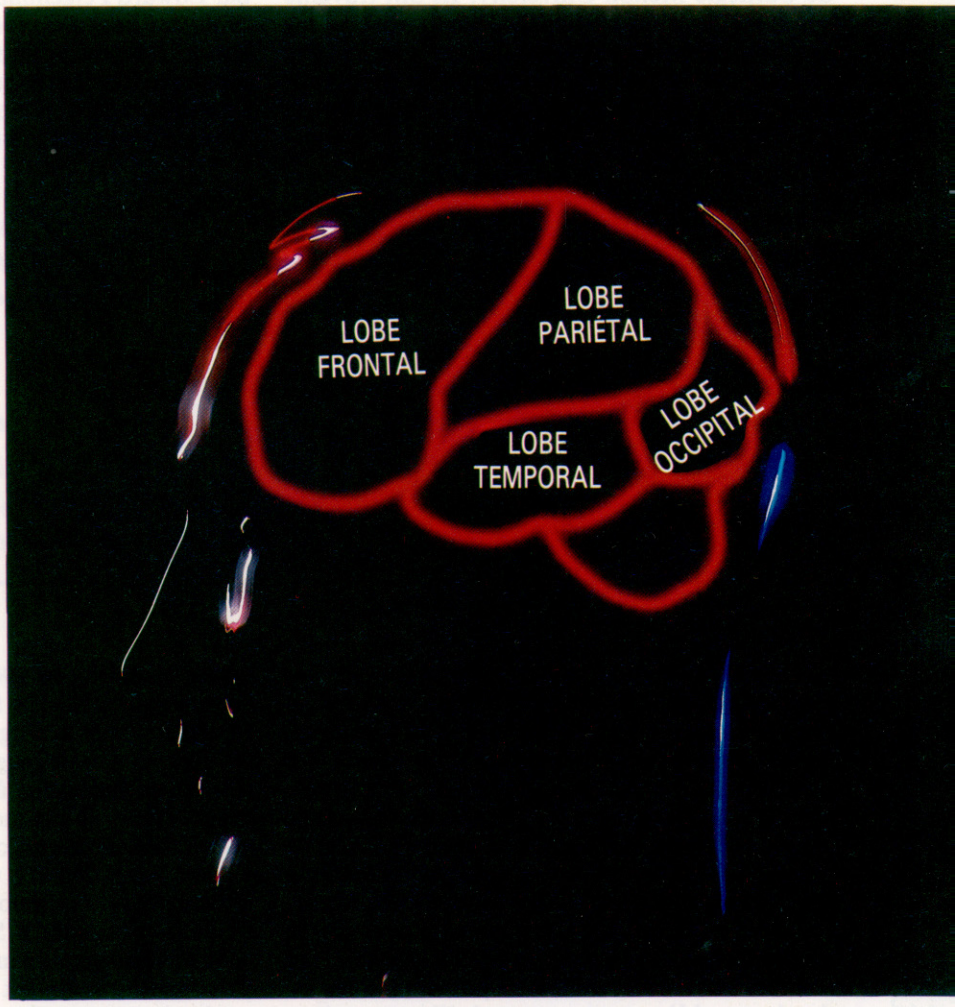
Au début des années soixante-dix, l'intelligence artificielle était en plein marasme. Aujourd'hui, ce domaine marche bien. Ses « adeptes » sont brusquement devenus tellement intéressants que les capitaux dits « à risque » n'hésitent plus à les financer, et leur courent pratiquement après. Les organismes gouvernementaux, quant à eux, soutiennent sans réserve de coûteux projets de recherche et de développement. Les divers pays impliqués par ce défi technologique n'ont qu'une peur : se laisser distancer et perdre pied dans cette course. Les éditeurs de logiciels s'efforcent pour leur part de redéfinir leurs produits en tant que systèmes faisant intervenir l'intelligence artificielle.

Pour comprendre où l'on en est et appréhender quel pourra être le futur, il est utile, là aussi, de regarder le passé. Un bref historique sur ce sujet dégagera quatre phases, sur quatre décennies. C'est bien sûr une simplification, mais elle a le mérite de souligner les points importants. Chaque thème ainsi mis en évidence répond en réalité à la même question : « Sur quoi portent les recherches en intelligence artificielle? »

1950 : réseaux neuraux.
1960 : recherche en heuristique.
1970 : systèmes experts.
1980 : acquisition automatique de connaissances.

En 1943, Warren McCulloch et Walter Pitts proposaient un modèle de neurones du cerveau humain et animal. Cette abstraction permit une représentation symbolique mathématique de l'activité cérébrale. D'autres chercheurs appliquèrent ce modèle et les idées s'y rapportant à un nouveau champ de recherche connu sous le nom de cybernétique. Ce système veut que l'on puisse construire une machine « intelligente » en partant de méthodes biologiques sur l'information en retour et l'analyse. La cybernétique a véritablement donné naissance à l'intelligence artificielle dans les années cinquante.

Les premiers chercheurs prirent le neurone fomalisé par McCulloch comme base. Vu l'immense complexité du cerveau, il est peu surprenant qu'ils aient échoué à créer un système intelligent sur ce modèle. Leur raisonnement, en effet, était le suivant : « Le cerveau est une machine qui résout des problèmes, aussi devons-nous simuler le cerveau. » Mais le matériel de l'époque, pour ne pas parler du logiciel, était loin d'être à la hauteur. Le système Perceptron de Rosenblatt fut une des rares réussites de l'époque. Il s'agissait d'un système visuel élémentaire, susceptible d'apprendre à reconnaître des formes. Le système Perceptron est constitué d'une fine grille de cellules photoélectriques, dessinant une rétine miniature. Il comprend en outre un certain nombre d'éléments qui lisent la disposition d'ensemble, et contrôlent l'état des groupes de cellules de la grille (graphiquement appelés « les démons »). Ils agissent, lorsque les éléments du motif sont en place, en envoyant un signal à un dispositif de décision. Ce dernier multiplie chaque signal en provenance d'un « démon », par un facteur positif ou négatif de pondération. Les nombres résultants sont enfin ajoutés. Lorsque le total dépasse un certain seuil, le Perceptron indique « oui »; dans le cas contraire, « non ». Ainsi la machine peut distinguer deux types d'images, mais le principe peut s'appliquer à plus



Esprit mécanique
La recherche en intelligence artificielle consiste à mettre au point des systèmes informatiques devant mener à bien des tâches qui demandent normalement une intelligence humaine. Son domaine d'application a été étendu au domaine de la perception, comme la vue et l'ouïe.
La préoccupation principale des recherches en ce domaine est de reproduire, à l'aide de la machine, le comportement humain et son mode de compréhension.
(Cl. Paul Chave.)

de deux. Le Perceptron allait-il résoudre de nombreux problèmes ? Ce ne fut pas le cas. Les chercheurs en intelligence artificielle conçurent alors la pensée humaine comme étant essentiellement la coordination de tâches simples et de traitements de symboles. C'était un changement radical dans la direction des recherches. Les concepteurs retrouvaient un terrain plus sûr, les ordinateurs pouvant gérer des symboles.

Les chercheurs les plus importants des années soixante furent Alan Newell et Herbert Simon, de Carnegie-Mellon University. Ils ont travaillé sur la vérification artificielle de théorèmes et sur l'ordinateur-joueur d'échecs. Leur plus grande réussite fut un programme appelé General Problem Solver (GPS), programme de résolution de problèmes d'ordre général. L'utilisateur définissait un environnement de travail, en termes correspondant aux objets d'un certain domaine et des opérateurs appropriés. Dans la pratique, néanmoins, cet environnement se limitait à des problèmes pouvant prendre un nombre limité d'états qui relevaient d'un nombre relativement petit de règles bien définies. Il pouvait traiter le puzzle dit des Tours de Hanoi. L'arithmétique des puzzles et énigmes, dans laquelle des micro-univers formalisés représentaient les paramètres de résolution, était en fait la seule à pouvoir être résolue. Par contre, le GPS ne pouvait pas résoudre les problèmes se posant à l'homme dans sa vie quotidienne.

Le principe de base de ce programme était que la résolution d'un problème revenait à effectuer une recherche à l'intérieur d'un champ limité de solutions potentielles. Pour optimiser la recherche effectuée par l'ordinateur, il fallait utiliser l'heuristique. Cette méthode consiste à apprendre, à partir des dernières découvertes qu'on a faites, à en tirer les conclusions qui permettent de diriger les recherches du programme dans une nouvelle direction. L'ordinateur pratique lui-même cette démarche. Le chemin logique est tracé par une suite d'essais et d'erreurs. De la sorte, un automate parcourant un labyrinthe mettrait en œuvre une technique de recherche complète, c'est-à-dire, qu'en l'absence d'éléments d'informations sur le labyrinthe, il devrait essayer toutes les solutions possibles, au hasard. Mais si cet automate pouvait obtenir une réponse à son dernier essai (comme dans le jeu où l'on « brûle » en s'approchant de l'objet caché), il parviendrait beaucoup plus vite à son objectif. Les chercheurs ont à cette époque mis au point plusieurs stratégies pour résoudre des problèmes par une approche heuristique.

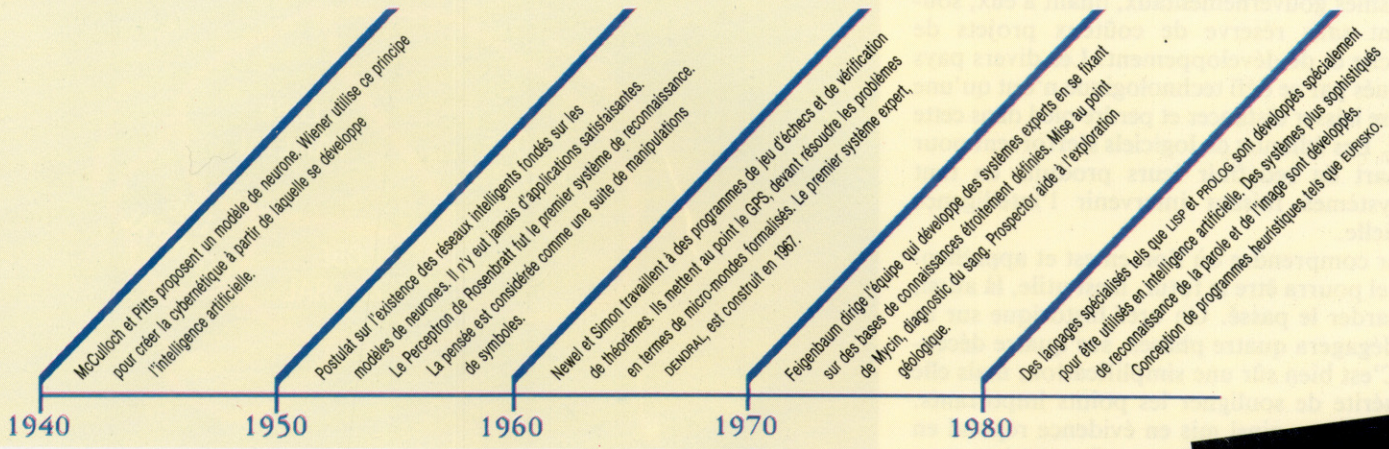
Le programme GPS, nous l'avons vu, n'est pas adapté aux problèmes les plus simples de la vie quotidienne. Au cours des années soixante-dix, une équipe de la Stanford University, conduite par Edward Feigenbaum, s'attaqua à cette situation. Au lieu d'essayer d'informatiser l'intelligence, ils concentrèrent leurs efforts dans des domaines très limités sur ce qui devait devenir des systèmes experts. Le premier système expert fut Dendral, un interpréteur de spectrogramme de masse construit en 1967. Néanmoins, celui qui eut le plus d'influence fut Mycin, réalisé en 1974. Mycin effectue le diagnostic d'infections bactériologiques dans le sang, et prescrit un traitement. Il a donné naissance à toute une généalogie de programmes de diagnostics, certains étant passés au stade d'utilisation courante en milieu hospitalier. Ainsi de Puff, un programme concernant les maladies du poumon, en service dans le Pacific Medical Center à San Francisco.

Mycin innova avec un certain nombre de caractéristiques qui allaient devenir les standards des systèmes experts. En premier lieu, ses connaissances consistaient en plusieurs centaines de règles, comme les suivantes :

RÈGLE N° 47.
SI 1) le site de culture est le sang, et
2) l'identité de l'organisme n'est pas connue avec certitude, et
3) coloration gram négatif, et
4) morphologie de l'organisme en bâtonnets, et
5) le patient est sérieusement brûlé
ALORS il y a peu de chances (0,4), que l'identité des organismes incriminés soit des pseudo-membranes.

En deuxième lieu, ces règles fonctionnent selon des degrés de probabilité. Shortliffe, l'inventeur de Mycin, était médecin. Son système était fondé sur les facteurs de probabilité permettant au pro-

Perspective historique
L'intelligence artificielle occupe une place croissante dans les recherches appliquées. Son histoire couvre cependant une période très courte en comparaison avec de nombreux autres domaines de la recherche scientifique, comme le montre notre diagramme.



(Liz Dixon)

Edward Feigenbaum
Chef d'une équipe de chercheurs en intelligence artificielle à Stanford University, California, Edward Feigenbaum développa le premier système expert. (Cl. Computer Weekly.)

gramme de parvenir à des conclusions plausibles à partir de faits incertains. Ainsi le nombre 0,4 n'est pas vraiment une probabilité, c'est un facteur d'appréciation. Ce qui est important, c'est le fait que Mycin et les systèmes analogues peuvent parvenir à des conclusions exactes à partir d'informations incomplètes et même partiellement fausses. Ces systèmes experts utilisent un mode de raisonnement par approximation basé soit sur les probabilités comme Fuzzy Logic dont nous avons déjà parlé, soit sur des facteurs de certitude ou autres calculs similaires, pour obtenir une estimation satisfaisante de la vérité à partir d'un ensemble imparfait de données.

En troisième lieu, Mycin peut expliquer son propre raisonnement. Le praticien qui l'utilise peut l'interroger de plusieurs manières, pour savoir comment il est arrivé à telle conclusion, ou encore pourquoi il demande telle information. Le système répond en retraçant la démarche qui l'a amené à ses conclusions, et décrit ses déductions étape par étape. Cette convivialité du programme découle tout naturellement des règles suivies pour former son style de programmation.

En dernier lieu, et c'est le point le plus important, Mycin est un programme qui « marche ». Il réalise des diagnostics qui demandent à l'homme un très haut niveau de qualification professionnelle. Il est néanmoins surtout utilisé pour l'enseignement.

Nous voici arrivés aux années quatre-vingt. Les systèmes experts sont à la mode, leur ingrédient déterminant est maintenant la « connaissance ». L'étendue et la qualité des connaissances d'un système déterminent directement son efficacité. Mais la « Connaissance » n'est pas quelque chose que l'on peut mettre dans un programme comme on remplit un bocal. Codifier le savoir et le savoir-faire d'un expert dans un domaine particulier de connaissances est une démarche longue et difficile. Aussi, pendant que le monde entier s'émerveille sur les systèmes experts, l'intelligence artificielle se tourne vers un autre domaine de recherche : comment la machine peut acquérir d'elle-même des connaissances. Le centre d'intérêt est aujourd'hui le programme Eurisko.

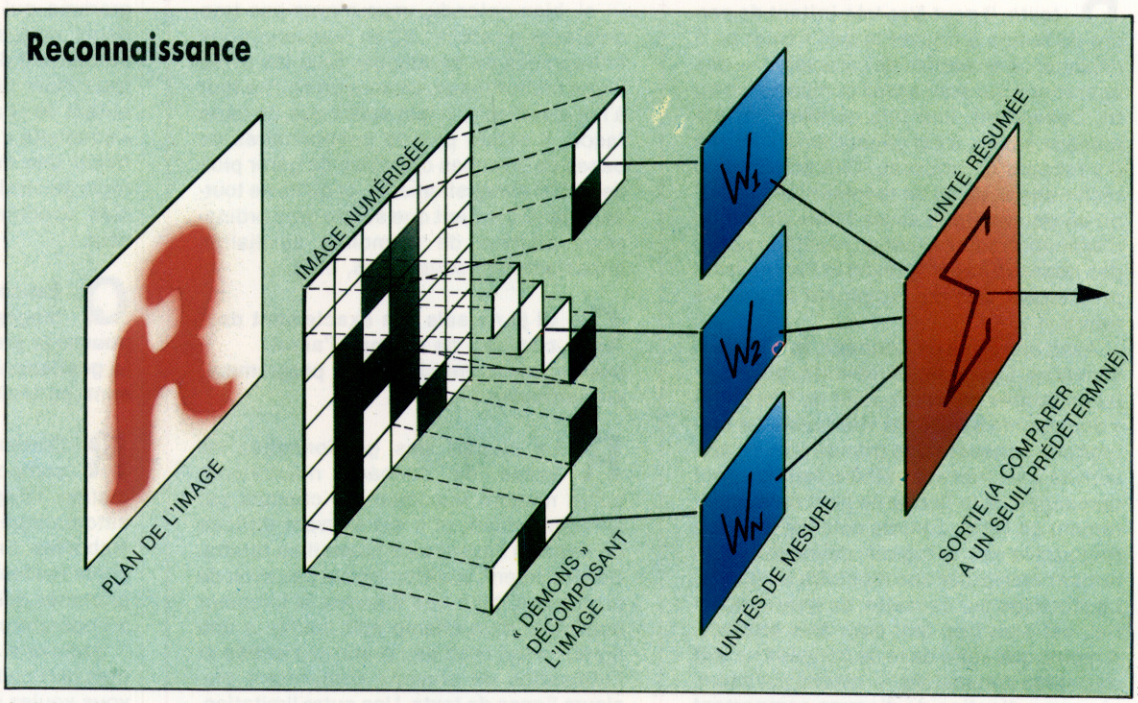
Il s'agit d'un programme d'exploration du champ des connaissances capable d'affiner automatiquement, par induction, ses propres règles heuristiques. Il s'applique tant à des jeux compliqués qu'à des problèmes pratiques. Parmi ses réalisations, l'invention d'une nouvelle porte logique à trois dimensions dans le domaine des circuits intégrés.

Il n'y a aucun doute que des systèmes comme Eurisko constituent l'avant-garde en matière de recherche en intelligence artificielle. Et comme cette dernière est également au premier plan du développement informatique, on peut mieux discerner l'avenir. Curieusement, en s'intéressant à nouveau à l'acquisition des connaissances, l'intelligence artificielle est revenue à ses racines. C'était en effet, à ses débuts, la préoccupation majeure de la cybernétique.

Cette suite d'articles représente un guide pratique de l'intelligence artificielle. Nous verrons ses principaux domaines (tels que le traitement de l'image et des langages naturels), ainsi que les techniques mises en œuvre. Comme toujours, tous les exemples de programmes sont en BASIC. Des encadrés présenteront les segments de programmes propres à certains micro-ordinateurs. Le chercheur en intelligence artificielle a son laboratoire rempli de grosses machines utilisant LISP. Mais il faudra attendre encore longtemps, probablement, pour voir ces systèmes sur micro-ordinateurs et interpréteurs BASIC. Nous espérons pouvoir néanmoins illustrer certaines techniques clés de l'intelligence artificielle, par des exemples destinés à votre machine, qui est, après tout, aussi puissante que les systèmes centraux d'il y a vingt ans.

Les perceptions de Perceptron

L'image à représenter (ici la lettre R) est projetée sur un plan et numérisée. Les « Démons » présentent de petits groupes de pixels (par exemple quatre à la fois), et répondent lorsque le schéma pour lequel ils sont programmés est reconnu. Leur réponse (0 ou 1), est multipliée par un facteur de correction, selon l'importance par rapport à l'ensemble, et toutes les réponses ainsi pondérées sont ajoutées les unes aux autres. Ce résultat est alors comparé à une valeur seuil : s'il excède cette dernière, le système a reconnu la forme originale; dans le cas contraire, il ne la reconnaît pas.



Le bon choix

Avant de faire l'acquisition d'un gestionnaire de base de données — qui n'est pas donné —, il vous faut prendre en compte un grand nombre d'éléments techniques.

Q Quand on veut faire l'achat d'un gestionnaire de base de données, quels sont les points à considérer en premier lieu ?

R Quand on se sert d'un traitement de texte, il peut être très irritant de perdre du temps à trouver le bon fichier ou à déplacer un ensemble de caractères — des opérations qui sont assez comparables au tri. Cependant, une fois qu'elles ont été menées à bien, il est généralement inutile d'y recourir de nouveau. Ces activités de recherche et de tri ne représentent en fait qu'un pourcentage assez faible du temps d'exécution total. C'est en réalité la saisie des données (taper le texte) et l'impression proprement dite qui réclament le plus de temps.

Avec une base de données, la situation s'inverse. Les enregistrements, pour parler en termes généraux, sont de très petits « documents » qui sont vite tapés. Le programme consacre une grande part de son activité à les trier et à extraire ceux qui sont demandés. Si le fichier ne peut tenir entièrement en RAM, comme cela est presque toujours le cas, le logiciel devra obligatoirement manipuler une mémoire auxiliaire. Le stockage sur cassette, de type séquentiel, est bien trop lent pour être efficace. A moins que votre base de données ne soit de petites dimensions — disons par exemple une collection de disques comportant une centaine de références —, vous aurez besoin d'un système basé sur disquette, ou du moins vous feriez mieux d'envisager cette éventualité.

Presque toutes les bases de données sont utilisées, d'une façon ou d'une autre, pour la gestion (« le temps c'est de l'argent »). Cela rend indispensable l'achat d'un programme sur disquette, encore qu'Archive, utilisé par les microdrives du QL, puisse être considéré comme une alternative intéressante.

Q Combien d'enregistrements mon gestionnaire de bases de données devrait-il pouvoir manipuler, et est-il intéressant d'en avoir un qui me propose plus d'espace que je n'en aurai jamais besoin ?

R Suivant ce que vous voulez faire, vous aurez besoin, ou non, de pouvoir manipuler un grand nombre d'enregistrements. Essayez d'évaluer le nombre le plus élevé imaginable; multipliez-le par deux, puis efforcez-vous de trouver un gestionnaire de bases de données qui soit capable de gérer ce chiffre. Si, par exemple, vous mettez sur pied une base de données consacrée à des stocks et à des inventaires, vous n'aurez sans doute pas l'occasion de dépasser les 32 000 enregistrements — si, bien entendu, vous n'avez pas trop de pièces en stock ! Si, en revanche, vous entreprenez de cataloguer tous les livres d'une bibliothèque, ce sera un chiffre tout à fait insuffisant. Il faut savoir que, de toute façon, il existe peu de gestionnaires de bases de données capables de gérer plus de 64 000 enregistrements, et qu'ils ne tournent généralement que sur de gros ordinateurs pourvus de mémoires auxiliaires importantes (disques durs, etc.).

Q Si je ne sais pas exactement de combien de champs j'aurai besoin, comment choisir un programme en particulier ?

R La plupart des gestionnaires de bases de données n'autorisent qu'un nombre maximum de champs par enregistrement, et il est souvent difficile de savoir à l'avance combien il en faudra. Généralement, vous ne devriez pas avoir de problèmes, bien que souvent la longueur maximale d'un champ soit limitée à une ligne, ce qui rend peu pratique l'entrée et l'utilisation de champs comportant plusieurs lignes de texte. Une autre limitation, bien plus importante, est qu'un enregistrement ne peut contenir d'ordinaire qu'un certain nombre de caractères : disons, pour prendre un exemple typique, 1 020. L'usager doit donc tenir compte, à la fois du nombre de champs et du nombre de caractères dans chacun d'eux, de façon à gérer au mieux l'espace dont il dispose. Une bonne méthode consiste à en définir, dans les deux cas, le plus grand nombre et à s'assurer qu'il permettra de satisfaire tous les besoins.

Q Les champs clés sont-ils vraiment si importants, et, si oui, combien m'en faudra-t-il ?

R Un champ clé peut être recherché, ou manipulé au sens large, par le gestionnaire de bases de données. Il faut, bien sûr, le désigner en tant que tel, sinon il ne pourra servir à l'extraction des données. Voici un ou deux exemples qui vous permettront de vous faire une idée :

SEARCH FOURNISSEUR FOR « Bourcier Management S.A. »

Le champ clé FOURNISSEUR de chaque enregistrement est alors analysé de façon à voir s'il contient la chaîne de caractères spécifiée. Ou encore :

SEARCH PRIX> = 45.50

Le champ clé PRIX est parcouru par le programme qui cherche des chiffres supérieurs, ou égaux, à 45.50. Certains logiciels permettent de définir tous les champs, ou tous ceux dont on aura besoin, comme autant de champs clés, dès la création du « squelette » de l'enregistrement; d'autres fixent une limite de ce point de vue. En règle générale, plus le nombre de champs clés spécifiables est élevé, mieux cela vaut.

Q Est-ce que je serai limité au langage de mon ordinateur, ou pourrai-je programmer moi-même le gestionnaire de bases de données, sans en tenir compte ?

R Certains programmes de ce type comportent un langage de programmation intégré, qui permet l'exécution automatique de séquences d'activité très élaborées. En gestion, des langages de ce genre rendent possible la rédaction de programmes qui épargneront bien des efforts à l'opérateur.

Archive et dBase II en sont deux exemples particulièrement impressionnants. Si vous voulez pouvoir procéder à des manipulations comme « imprimer toutes les ventes d'aujourd'hui, puis la liste de toutes les pièces en stock, et celles des pièces qui sont en dessous du seuil de « réassortiment », il sera judicieux de faire l'acquisition d'un logiciel disposant d'un véritable langage de programmation. Si, par contre, vous ne savez pas de façon précise quel usage vous ferez de votre base de données, un simple programme « commandes » devrait vous suffire.

Q Quels sont les points à considérer si les enregistrements que je crée doivent être reliés à d'autres données que celles qui y seront contenues ?

R Les gestionnaires de bases de données les plus simples ne peuvent généralement travailler que sur un seul fichier à la fois. Cela suffit parfaitement dans la plupart des cas. Il est pourtant très utile de pouvoir manipuler simultanément deux fichiers ou plus. Imaginons par exemple qu'un fichier de pièces en stock com-

porte des pièces ayant plus d'un fournisseur. Nous aurions un fichier « pièces », et un fichier « fournisseurs » séparé. Il est extrêmement pratique, en cas d'interrogation de l'un, de pouvoir également interroger l'autre.

Supposons que vous êtes antiquaire et que vous avez créé un fichier rassemblant tout ce que vous avez en stock dans votre magasin. Il serait intéressant de créer un fichier consacré à vos fournisseurs, qui préciserait tout ce qu'ils ont à vendre, même si vous ne leur avez pas encore acheté tel ou tel article. Vous pourrez très bien avoir un champ DESCRIPTION « MEUBLES SUÉDOIS » tout à fait vide, mais une consultation de votre fichier FOURNISSEURS vous permettrait de retrouver KURT JAKOBSEN ANTIK, GAMLA STAN 56, STOCKHOLM. L'enregistrement serait le suivant :

DESCRIPTION	SOFA
FABRICANT	ANDERSEN
DATE	1824
PRIX (COUR. SUÉD.)	12 000
DESCRIPTION	SOFA
FABRICANT	GRIMM
DATE	1874
PRIX (COUR. SUÉD.)	9 800

et d'autres entrées, non limitées en nombre, consacrées à tous les articles proposés par ce fournisseur. En d'autres termes, quand un seul fichier ne peut suffire, la référence à un ou plusieurs autres peut souvent vous tirer d'embarras.

Théoriquement, il est bien entendu toujours possible de créer suffisamment de champs dans un fichier particulier, de telle sorte qu'on puisse y intégrer toute information supplémentaire dont on aura besoin. Mais dans les faits, ce n'est pas toujours réalisable. Supposons que M. Jakobsen prenne la peine de nous téléphoner pour nous dire qu'il vient tout juste de recevoir deux charmants sofas du XIX^e^ qui pourraient peut-être nous intéresser. Il sera évidemment bien plus simple d'incorporer ce genre de détails dans l'enregistrement consacré à JAKOBSEN, plutôt que de parcourir tous les champs DESCRIPTION de notre fichier PIÈCES pour y entrer les nouvelles acquisitions de notre fournisseur.

Pour simplifier, disons que chaque fois qu'il existe une relation non univoque entre les enregistrements d'un fichier et d'autres données, il faut d'abord considérer en premier lieu un gestionnaire de bases de données multifichiers.

Q Les champs « dépendants » ou « calculés » semblent très utiles. Quel usage puis-je en faire quand je travaille avec une base de données ?

R Les champs « dépendants » constituent un raffinement très intéressant. On appelle ainsi des champs qui n'apparaissent (au moment de la saisie des données, et ultérieurement) que si l'on a besoin d'eux. C'est ainsi qu'une base de données comportant un champ intitulé NOMBRE D'ENFANTS ne comportera pas d'autres champs si le chiffre en question est 0. En revanche, s'il est de 1, on verra apparaître NOM DU PREMIER ENFANT, ÂGE DU PREMIER ENFANT, et ainsi de suite.

Comme tous les raffinements, ce genre de précision ne se révèle que très rarement utile. Certains programmes permettent également d'employer les données contenues dans certains champs comme autant d'arguments mis en œuvre dans des opérations arithmétiques, un peu comme un minitableur. C'est ainsi que le nombre d'articles dans un champ NOMBRE EN STOCK pourrait être multiplié par le chiffre contenu dans le champ PRIX de chaque enregistrement, tous les résultats étant stockés dans un fichier séparé, de façon à donner une valeur totale du stock. Ce genre de possibilité, qui peut être très profitable, est généralement lié à la présence d'un langage de programmation intégré.

Q Quelles sont les différences entre les systèmes gérés par menus, et ceux gérés par commandes ? Quels sont les plus performants ?

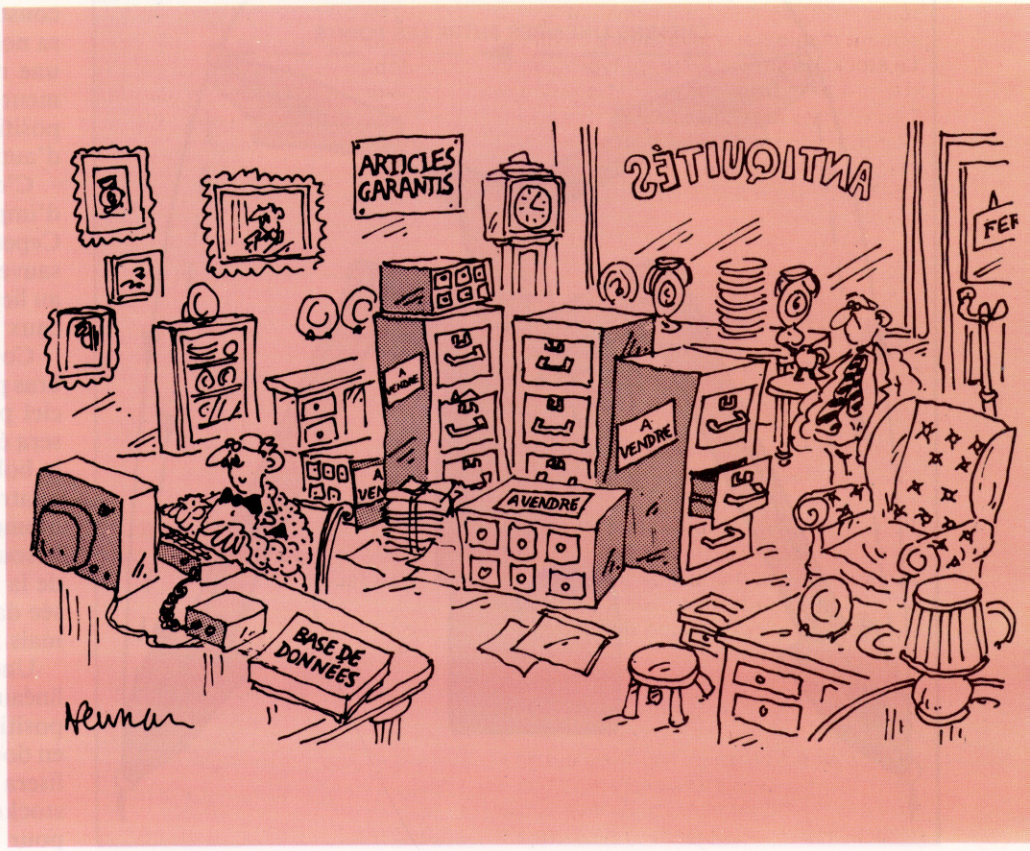
R Une base de données gérée par menus vous présente un choix d'options à chaque étape du traitement de l'information. Pour vous servir d'un système à commandes, vous aurez besoin de mémoriser tout un vocabulaire (ce qui se réduit en fait à l'apprentissage de combinaisons touche CTRL + une série d'autres touches), avant de pouvoir vous mettre au travail. Les systèmes à menus sont d'usage plus facile pour le débutant, mais les autres sont généralement plus rapides, dès lors que vous avez appris les commandes. Ils sont généralement plus flexibles et plus faciles de maniement à long terme, pourvu évidemment que les commandes aient été conçues selon une certaine logique (P pour PRINT, S pour SEARCH, etc.), ce qui n'est pas toujours le cas.

Q Quelles sont les ressources dont disposent les gestionnaires de bases de données pour détecter toute erreur éventuelle, et empêcher toute saisie de données erronées ?

R Un système bien conçu devrait vous permettre de préciser quel type de données vous pouvez intégrer dans chaque champ, ce qui vous aide à prévenir toute erreur. Par exemple 72/02/85 se verrait automatiquement rejeté, de même que PRIX : manomètre à pression. Les meilleurs programmes disposent toujours d'une possibilité de définir des limites relatives au type de données acceptables. D'un autre côté, il faut bien voir qu'un système de vérification trop rigide se révèle souvent très gênant. La solution la plus satisfaisante consiste en fait à voir quelles sont réellement les possibilités de contrôle de la validité des données.

Q Et maintenant, quel programme dois-je acheter ?

R Il est absolument impossible de recommander un logiciel plutôt qu'un autre. Vous devrez d'abord savoir à quoi vous pourrez l'utiliser avant de l'acheter ! Nous nous bornerons à vous recommander — sauf si vous vous limitez à quelques applications très simples — un système à disquette, multifichier, et qui dispose d'un langage de programmation intégré.



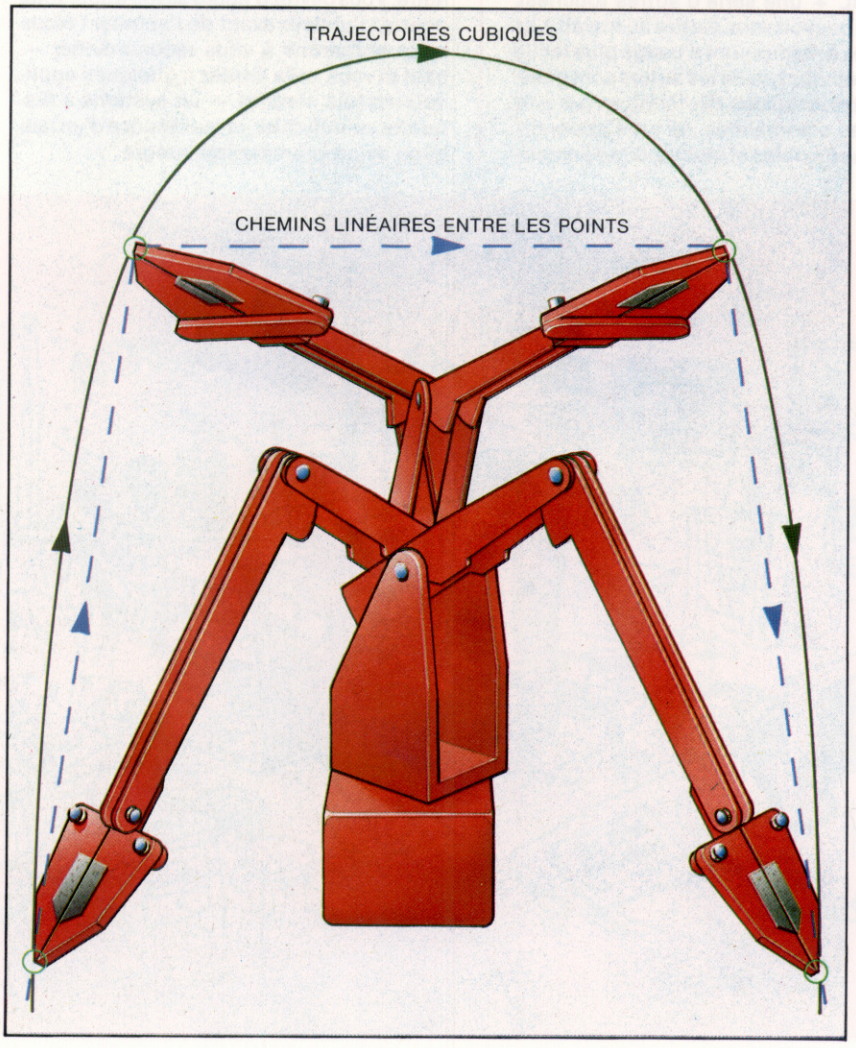
Curviligne

Nous allons modifier le logiciel mis au point pour notre bras-robot. Nous verrons également des mises au point pour améliorer la régularité des mouvements du bras.

Adoucir le mouvement
Si quatre positions limites du bras sont sauvegardées lors de l'utilisation du programme du bras, ce dernier se déplacera linéairement entre ces points (pointillés) lorsque la séquence est réexécutée. Cependant, il est possible de calculer de nombreux points intermédiaires qui génèrent une trajectoire curviligne (dite trajectoire cubique) passant par les quatre positions sauvegardées.
(Cl. Kevin Jones.)

La routine moveservo, mise au point dans le logiciel de commande du bras, réduit les secousses produites lors d'un déplacement d'une position à l'autre. Les moteurs reçoivent leur information angulaire à partir d'une série d'emplacements en mémoire, commençant à ANGLE.

Pour améliorer la régularité du mouvement du bras, lorsque les moteurs adoptent de nouvelles positions en réponse aux changements apportés dans les emplacements ANGLE, les valeurs ne sont pas écrites (POKE) directement dans ces emplacements mais dans un groupe d'emplacements correspondants nommés NEWPOS. La routine moveservo, lorsqu'elle est appelée, compare les valeurs des emplacements correspondants de ANGLE et de NEWPOS et diminue ou augmente ANGLE de façon appropriée, une unité à la fois. Elle répète ce processus jusqu'à ce que les valeurs de NEWPOS et de ANGLE soient identiques.



En écrivant les valeurs d'angle dans NEWPOS et non dans ANGLE, nous créons en fait un tampon qui prévient tout changement brusque et important de l'angle des moteurs. La tâche consistant à prendre une bobine de fil à une certaine position et à la déposer dans une tasse est maintenant relativement facile. Le bras peut se mouvoir lentement à l'aide des touches appropriées du clavier, et les positions clés de la courbe peuvent être sauvegardées dans un tableau. L'information contenue dans le tableau peut être réexécutée à une vitesse précise, en insérant un facteur de retard dans moveservo et en parcourant le tableau pas à pas.

Il est possible d'apporter d'autres améliorations. Une première approximation nous apprend que les mouvements angulaires sont proportionnels aux changements effectués dans NEWPOS, selon la distance séparant chaque liaison du pivot comparée avec le rayon du levier du moteur. La routine moveservo fait donc bouger uniformément chaque partie du bras à une vitesse angulaire constante (degrés de rotation par seconde) vers sa nouvelle position. Cependant, lorsqu'il atteint une nouvelle position, le bras change brusquement de direction en commençant à gagner la position sauvegardée suivante de la séquence; en d'autres mots, le mouvement est saccadé.

C'est déjà mieux que d'être contraint à effectuer d'importants et soudains changements d'angle. Cependant, ce problème peut être contourné en sauvegardant de nombreux points intermédiaires au lieu d'avoir à introduire des changements radicaux entre deux positions sauvegardées.

Comme c'est souvent le cas, le mouvement du bras peut être encore amélioré en utilisant un logiciel plus perfectionné, mais toute amélioration sera évidemment coûteuse en espace mémoire. Si la bobine de fil est déplacée en ne précisant que quatre positions distinctes — la première au moment de la saisie de la bobine, la deuxième en la soulevant, la troisième en la plaçant au-dessus de la tasse, et la quatrième lorsqu'elle est déposée dans la tasse — peu de mémoire est utilisée, mais le mouvement est très saccadé.

Une manière de rendre ce mouvement plus linéaire consiste à spécifier, par exemple, soixante positions intermédiaires permettant d'effectuer en douceur une trajectoire élégante. Mais cela utilisera une quantité importante de mémoire pour stocker les données, sans parler du temps requis pour entrer manuellement toutes les positions.

Il serait également possible de calculer de nombreuses valeurs intermédiaires et de combiner les positions qu'elles représentent avec celles que vous avez sauvegardées manuellement.

Le programme donné ici démontre comment des positions intermédiaires qui correspondent à une courbe cubique peuvent être interpolées à partir des quatre points particuliers. Le mouvement résultant entre les points suivra une trajectoire courbe régulière et non saccadée ; c'est une amélioration importante par rapport à l'interpolation linéaire. Pour utiliser cette méthode avec un système comportant quatre moteurs, il est nécessaire d'interpoler mathématiquement une trajectoire cubique pour chaque moteur.

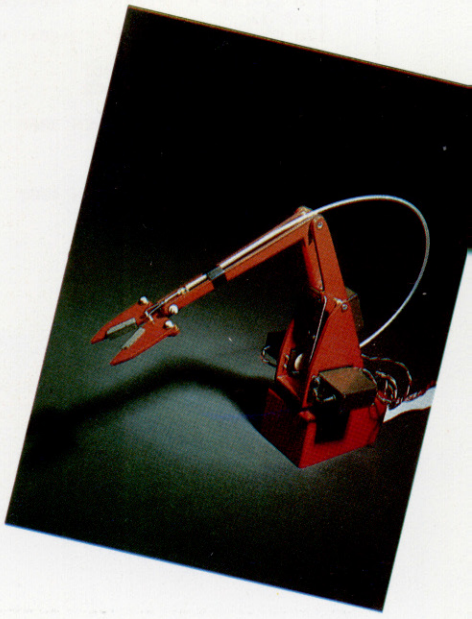
Trajectoires cubiques

Spectrum

```
1000 REM ******************************************
1010 REM **  Trajectoires Cubiques du Spectrum **
1020 REM ******************************************
1030 :
1060 V=7
1070 DIM X(V+3),Y(V+3),M(V+3),Z(V+3)
1080 X(2)=0 :X(3)=10 :X(4)=40 :X(5)=50
1090 Y(2)=2 :Y(3)=12 :Y(4)=12 :Y(5)=2
1100 REM étendre les extrémités linéairement
1110 X(1)=X(2)-(X(3)-X(2))
1120 Y(1)=Y(2)-(Y(3)-Y(2))
1130 X(6)=X(5)+(X(5)-X(4))
1140 X(7)=X(6)+(X(6)-X(5))
1150 Y(6)=Y(5)+(Y(5)-Y(4))
1160 Y(7)=Y(6)+(Y(6)-Y(5))
1170 CLS
1180 PRINT"Démonstration d'une routine"
1190 PRINT"qui intègre une trajectoire cubique"
1200 PRINT"en joignant les points intermédiaires"
1210 FOR I=2 TO 5
1220 CIRCLE X(i)*5-4,Y(i)*5+15,10
1230 NEXT
1240 GOSUB 1500:REM DEFINIR FONCTION AKIMA
1250 FOR X=X(2) TO X(5)
1260 GOSUB 1330:REM FONCTION AKIMA
1270 IF X=X(2) THEN PLOT X*5,Y*5
1280 IF X<>X(2) THEN DRAW X*5,4*5
1290 NEXT
1300 END
1310 :
1320 :
1330 REM **** FONCTION AKIMA ****
1340 REM AKIMA RUCKDESCHEL BOOK 2  (BYTE/ MCGRAW-HILL)
1350 N=1
1360 REM VERIFIER SI X DANS LIMITES DE LA TABLE
1370 IF X<X(1) OR X>X(V-2) THEN RETURN
1380 REM TROUVER INTERVALLE DE TABLE CORRESPONDANT
1390 I=0
1400 I=I+1:IF X>=X(I) THEN 1400
1410 I=I-1
1420 REM COMMENCER INTERPOLATION
1430 B=X(I+1)-X(I)
1440 A=X-X(I)
1450 Y=Y(I)+Z(I)*A+(3*M(I+2)-2*Z(I)-Z(I+1))*A*/B
1460 Y=Y+(Z(I)+Z(I+1)-2*M(I+2))*A*A*A/(B*B)
1470 RETURN
1480 :
1490 :
1500 REM **** DEFINIR AKIMA ****
1510 REM DEFINIR COEF AKIMA
1520 FOR I=1 TO V-1
1530 REM SHIFT I TO I+2
1540 M(I+2)=(Y(I+1)-Y(I))/(X(I+1)-X(I))
1550 NEXT I
1560 M(V+2)=2*M(V+1)-M(V)
1570 M(V+3)=2*M(V+2)-M(V+1)
1580 M(2)=2*M(3)-M(4)
1590 M(1)=2*M(2)-M(3)
1600 FOR I=1 TO V
1610 A=ABS(M(I+3)-M(I+2))
1620 B=ABS(M(I+1)-M(I))
1630 IF A+B<>0 THEN Z(I)=(A*M(I+1)+B*M(I+2)/(A+B)
1640 IF A+B=0 THEN Z(I)=(M(I+2)+M(I+1))/2
1650 NEXT I
1660 RETURN
```

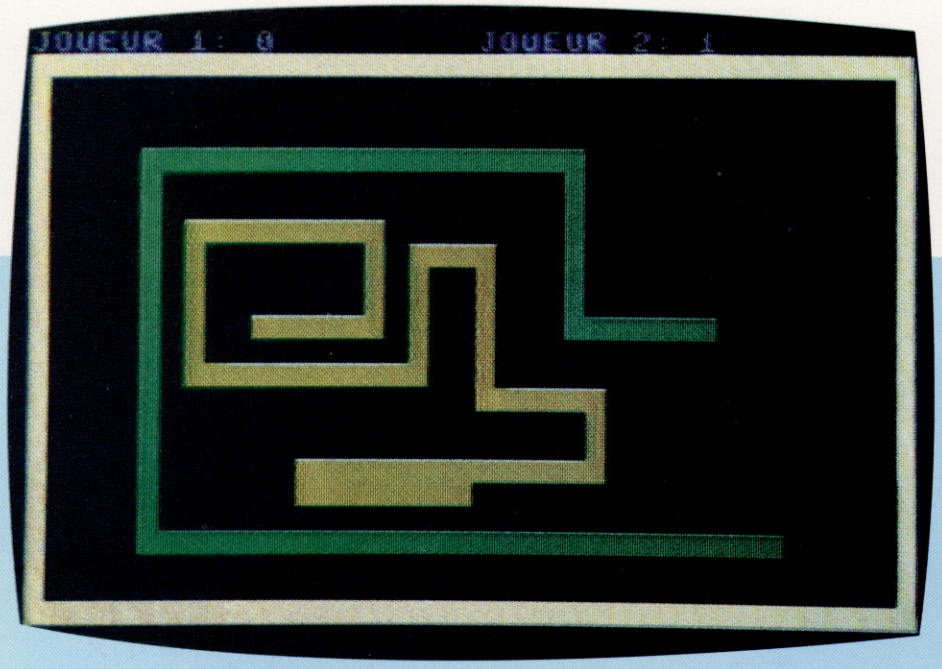
Commodore 64

```
10 REM ********************************
20 REM **  TRAJECTOIRES CUBIQUES  **
30 REM ********************************
40 :
50 V=7:FOR I=1 TO 25:DW$=DW$+CHR$(17):NEXT
60 DIM X(V+3),Y(V+3),M(V+3),Z(V+3)
70 X(2)=0:X(3)=5:X(4)=20:X(5)=25
80 Y(2)=2:Y(3)=12:Y(4)=12:Y(5)=2
90 REM ** ETENDRE LINEAIREMENT LES DEUX EXTREMITES **
100 X(1)=X(2)-(X(3)-X(2))
110 Y(1)=Y(2)-(Y(3)-Y(2))
120 X(6)=X(5)+(X(5)-X(4))
130 X(7)=X(6)+(X(6)-X(5))
140 Y(6)=Y(5)+(Y(5)-Y(4))
150 Y(7)=Y(6)+(Y(6)-Y(5))
160 PRINTCHR$(147)
170 PRINT "DEMONSTRATION D'UNE ROUTINE"
180 PRINT "QUI INTEGRE UNE TRAJECTOIRE CUBIQUE"
190 PRINT "JOIGNANT DES POINTS"
200 GOSUB 3000:REM DESSINER DES POINTS
230 GOSUB 1000:REM DEFINIR AKIMA
235 J=2
240 FOR X=X(2) TO X(5)
250 GOSUB2000:REM FONCTION AKIMA
255 IF X=X(J) THEN J=J+1:GOTO270:REM DON'T DRAW
260 N=X:M=Y:GOSUB5000:PRINT CHR$(46)
270 NEXT
280 GET A$:IF A$="" THEN 280
290 END
1000 REM **** DEFINIR AKIMA ****
1010 REM CALC AKIMA COEFS
1020 FOR I=1 TO V-1
1030 M(I+2)=(Y(I+1)-Y(I))/(X(I+1)-X(I))
1040 NEXT I
1050 M(V+2)=2*M(V+1)-M(V)
1060 M(V+3)=2*M(V+2)-M(V+1)
1070 M(2)=2*M(3)-M(4)
1080 M(1)=2*M(2)-M(3)
1090 FOR I=1 TO V
1100 A=ABS(M(I+3)-M(I+2))
1110 B=ABS(M(I+1)-M(I))
1120 IF A+B<>0 THEN Z(I)=(A*M(I+1)+B*M(I+2))/(A+B)
1130 IF A+B=0 THEN Z(I)=(M(I+2)+M(I+1))/2
1140 NEXT I
1150 RETURN
2000 REM **** FONCTION AKIMA ****
2010 N=1
2020 IF X<X(1) OR X>X(V-2) THEN RETURN:REM CHECK RANGE
2030 I=0
2040 I=I+1:IF X>=X(I) THEN 2040
2050 I=I-1
2060 REM COMMENCER INTERPOLATION
2070 B=X(I+1)-X(I)
2080 A=X-X(I)
2090 Y=Y(I)+Z(I)*A+(3*M(I+2)-2*Z(I)-Z(I+1))*A*A/B
2100 Y=Y+(Z(I)+Z(I+1)-2*M(I+2)*A*A*A/(B*B)
2110 RETURN
3000 REM **** IMPRIMER POINTS ****
3010 FOR I=2 TO 5
3020 N=X(I):M=Y(I):GOSUB 5000:PRINT CHR$(215)
3030 NEXT I
3040 RETURN
5000 REM **** POSITION N,M ****
5010 PRINT CHR$(19);
5020 PRINT TAB(N);LEFT$(DW$,25-M);
5030 RETURN
```



Trace sur C64

Qui ne connaît pas ce jeu? Voilà une bonne occasion pour les possesseurs d'un micro C64 de le traduire en BASIC et de remplacer la feuille par l'écran.



Deux joueurs s'affrontent pour se partager l'espace vital. Chacun doit s'efforcer, tout en se déplaçant, de ne jamais recouper sa trace ou celle de son adversaire, et de ne pas sortir du rectangle dessiné sur l'écran. Les commandes à utiliser sont :

Joueur de droite : <P>, <L>, <:> et <.>
Joueur de gauche : <W>, <A>, <S> et <Z>.

```
5 REM  **********
10 REM * TRACE *
15 REM **********
20 GOSUB 1000
100 GET X$
110 C1=(X$="A")-(X$="S")+40*((X$="W")-(X
$="Z"))
120 C2=(X$="L")-(X$=":")+40*((X$="P")-(X
$="."))
130 IF C1<>0 THEN D1=C1
140 IF C2<>0 THEN D2=C2
150 P1=P1+D1
160 IF PEEK(P1)<>32 THEN 3000
170 POKE P1,C
180 POKE P1+M,K1
190 P2=P2+D2
200 IF PEEK(P2)<>32 THEN 4000
210 POKE P2,C
220 POKE P2+M,K2
230 FOR I=1 TO 50
240 NEXT I
250 GOTO 100
1000 P1=1514
1010 P2=1534
1020 K1=7
1030 K2=5
1040 C=160
1050 M=54272
1060 D1=1
1070 D2=-1
2000 PRINT CHR$(147);
2010 POKE 53280,0
2020 POKE 53281,0
2030 FOR I=0 TO 39
2040 POKE 1064+I,C
2050 POKE 1064+I+M,1
2060 POKE 1984+I,C
2070 POKE 1984+I+M,1
2080 NEXT I
2090 FOR I=1 TO 22
2100 POKE 1064+I*40,C
2110 POKE 1064+I*40+M,1
2120 POKE 1103+I*40,C
2130 POKE 1103+I*40+M,1
2140 NEXT I
2160 POKE P1,C
2170 POKE P1+M,K1
2180 POKE P2,C
2190 POKE P2+M,K2
2200 PRINT "JOUEUR 1:";J1,"JOUEUR 2:";J2

2210 RETURN
3000 J2=J2+1
3010 FOR I=1 TO 500
3020 GET X$
3030 NEXT I
3040 IF J2=10 THEN 5000
3050 GOTO 20
4000 J1=J1+1
4010 FOR I=1 TO 500
4020 GET X$
4030 NEXT I
4040 IF J1=10 THEN 5000
4050 GOTO 20
5000 PRINT CHR$(147);
5010 FOR I=1 TO 18
5020 PRINT
5030 NEXT I
5040 PRINT TAB(10)"LE JOUEUR ";
5050 IF J1>J2 THEN PRINT "1";
5060 IF J2>J1 THEN PRINT "2";
5070 PRINT " GAGNE !"
5080 PRINT
5090 PRINT TAB(13)J1;"A";J2
5100 FOR I=1 TO 4
5110 PRINT
5120 GET X$
5130 NEXT I
5140 PRINT TAB(13)"UNE AUTRE ?"
5150 GET X$
5160 IF X$="" THEN 5150
5170 IF X$<>"N" THEN RUN
5180 END
```

Quand la Chine...

Une équipe de linguistes de l'université de Paris VII[e] a développé, en collaboration avec des informaticiens, un système de saisie et de traitement de texte chinois.

Terminal de saisie d'écriture chinoise ALFA.

Au moment où la Chine commence à développer ses échanges avec l'Occident et à entrer, à son tour, dans le monde de l'informatique et de la télématique, un obstacle demeure pour le peuple chinois : la complexité de son écriture.

Comment, à partir d'un clavier, saisir les quelques milliers de caractères les plus utilisés dans la langue chinoise? Jusqu'à présent, deux systèmes ont été envisagés, mais aucun n'est entièrement satisfaisant : il s'agit de l'entrée phonétique et de la décomposition graphique des idéogrammes.

Si le chinois est la langue unique en Chine, dans la mesure où, d'est à l'ouest, du nord au sud, tous les Chinois communiquent entre eux avec les mêmes caractères, il existe cependant énormément de dialectes : le même idéogramme se prononce d'une façon différente selon qu'on se trouve à Pékin, à Shanghai ou à Canton. De plus, à l'intérieur d'un même dialecte, il existe de nombreuses homophonies (des idéogrammes différents peuvent avoir la même prononciation), de sorte que, lorsqu'ils ne sont pas sûrs d'être compris, les Chinois ont l'habitude d'écrire les mots ambigus avec l'index sur la paume de leur main.

Pour lever l'ambiguïté, on peut, bien sûr, envisager d'afficher, pour chaque mot prononcé, la liste des caractères correspondants, l'usager n'ayant alors plus qu'à choisir l'un des homophones parmi les caractères affichés. Mais cette solution s'avère assez complexe et requiert de la part de l'opérateur plusieurs manipulations et beaucoup de temps.

Dans cette catégorie, on distingue la technique faisant appel à la reconnaissance vocale, où les mots chinois sont directement prononcés. Mais celle-là n'est pas parfaitement maîtrisée et exige,

Pour les Chinois, la saisie des données en informatique fut longtemps un véritable casse-tête... La reconnaissance dynamique, en temps réel, a été choisie pour la lecture des caractères écrits à la main, en style Kaishu.

de plus, que l'utilisateur prononce une première fois tous les mots qu'il utilisera, dans une phase préalable d'apprentissage. L'autre approche consiste à entrer les mots sous forme alphabétique, ce qui requiert de la part de l'opérateur l'apprentissage de l'écriture latine.

Le second procédé de saisie de caractères chinois consiste à utiliser les éléments graphiques des caractères : ce sont les « clés » — on distingue la clé de l'eau, celle du bois, du feu, de la main... —, qui composent les idéogrammes. Avec les 256 radicaux les plus usuels, ainsi que leur indication topologique (position de ces éléments : en haut, en bas, à gauche, à droite, au milieu...), il est possible de réaliser des claviers pouvant saisir la plupart des caractères chinois les plus usuels.

Les firmes Olympia, Wang, Monotype, etc., déjà implantées en Chine, ont adopté l'un ou l'autre de ces systèmes.

L'Association linguistique franco-asiatique (ALFA), association de type loi de 1901, qui regroupe des linguistes de l'université de Paris VII (Jussieu), des informaticiens du CRIN (Centre de recherche informatique de Nancy) et des spécialistes de l'automation, s'est penchée depuis 1979 sur la réalisation d'un terminal de saisie automatique et de traitement des caractères chinois, en vue d'une application dans le cadre de la coopération avec la Chine.

Certains de ces chercheurs avaient déjà travaillé sur un projet de terminal en langue arabe. Par ailleurs, des études avaient été entamées avec Taiwan. La première tâche de l'équipe consistait donc à établir un recensement de tout ce qui avait été fait en la matière, en particulier un dépouillement bibliographique considérable, impliquant l'abonnement à quelque cent cinquante revues.

La seconde étape consistait à choisir la méthode la mieux adaptée, avant d'aborder la phase finale : convaincre que le système était faisable, afin de déboucher sur un programme de recherche et de développement d'un tel terminal.

Il fallut aussi examiner les possibilités du marché chinois, afin de bien cerner la demande qui existait dans ce pays. Ce fut l'objet, pour des membres de l'ALFA, de plusieurs séjours en Chine, notamment six semaines au cours de l'été 1981.

Les recherches proprement dites commencèrent par la comparaison des avantages et inconvénients des différentes méthodes existantes. La méthode phonétique présentait les difficultés que nous avons déjà citées. Quant à la seconde méthode, son principal handicap est d'obliger à décortiquer le caractère, ce qui nécessite de la part de l'opérateur un processus mental non naturel.

L'ALFA a donc adopté un système reposant sur une tout autre approche : la reconnaissance graphique. Il ne requiert aucun apprentissage préalable de la part des utilisateurs et s'intègre parfaitement dans la culture chinoise.

Dès lors, le choix restait ouvert entre deux méthodes. La première, la lecture optique, a déjà été envisagée par les Japonais, mais ceux-ci se sont heurtés à des problèmes. La seconde, la reconnaissance dynamique, en temps réel, paraît être la mieux adaptée à la lecture des caractères écrits à la main, en style *kaishu*. C'est l'écriture chinoise standard.

C'est cette dernière approche qui a été retenue par l'ALFA pour son terminal baptisé « Lotus ». L'opérateur n'a qu'à écrire normalement son texte sur une feuille de papier posée sur une tablette à numériser, à l'aide d'un stylo spécial, relié au terminal. Le logiciel, qui s'appuie sur les programmes de reconnaissance des formes, identifie le tracé de l'écriture.

Dans un premier temps, tous les caractères doivent être mémorisés, mais cette opération s'effectue une fois pour toutes, indépendamment du futur utilisateur — contrairement à ce qui se passe habituellement en reconnaissance vocale. Après avoir été écrit sur la tablette, chaque caractère est filtré, lissé, normalisé, puis comparé à la forme du caractère mémorisé. Une fois qu'il a été reconnu, il est validé et l'idéogramme s'affiche sur l'écran du terminal, ce qui permet à l'opérateur de vérifier s'il a été bien compris.

Il arrive cependant qu'un même caractère puisse avoir deux ou trois variantes d'écriture. Dans ce cas, elles sont toutes mémorisées et peuvent être utilisées indifféremment.

Deux caractéristiques de l'écriture chinoise ont favorisé cette démarche : d'une part, les caractères sont indépendants les uns des autres (ils ne sont pas liés et leur graphisme ne dépend pas de ceux qui les précèdent ou les suivent); d'autre part, la façon de les écrire est rigoureuse — elle obéit à des séquences strictes. Toutefois, une certaine tolérance est admise sur la direction et la longueur des segments composant les idéogrammes.

Lotus utilise la norme chinoise de communication GB 23-12, qui équivaut à peu près à notre code ASCII. A chaque signe, elle fait correspondre deux octets. Sous cette forme codée, les caractères peuvent être traités normalement par l'informatique.

Outre la tablette à numériser, le terminal comporte un clavier numérique permettant d'entrer les chiffres, ainsi que les codes des caractères moins employés et dont le tracé n'est pas reconnu par le logiciel. Lotus peut également affecter à chaque caractère traité son code télex à quatre chiffres, et servir ainsi à éditer la bande perforée.

L'idée de l'ALFA est de faire de Lotus un périphérique adaptable à tout système; il recouvre tout le domaine des communications télex pour les entreprises, les administrations, les agences de presse, les journaux... Alors que, pour envoyer un télégramme en Chine, chaque caractère doit être converti manuellement en un code à quatre chiffres, à l'aide d'un index qui en comprend plusieurs milliers, puis reconverti, au bureau de poste de destination, en caractères chinois, l'automatisation de cette double opération de conversion permettrait d'économiser un temps précieux — souvent 48 heures pour un seul télégramme. En outre, elle garantirait la confidentialité des messages, avantage essentiel dans le domaine militaire, notamment.

Un tel système permettrait à tout le réseau de communications de faire un grand bond en avant et favoriserait par là même l'essor économique du pays et ses échanges avec l'étranger.

Aujourd'hui, la phase de recherche se termine, ouvrant la voie aux phases d'industrialisation et de commercialisation. Des contacts ont été établis avec des industriels français déjà implantés en Chine ou qui envisagent des rapports avec ce pays. Ce projet peut être proposé soit sous forme de coopération franco-chinoise, soit sous forme de capital-risque.

Selon Philippe Kantor, chercheur de l'ALFA, qui a séjourné à plusieurs reprises en Chine dans le cadre de cette étude, la première solution serait plus motivante pour les Chinois et peut-être plus adaptée à leurs besoins, car ils pourraient apporter une contribution précieuse, en particulier pour l'affichage des caractères.

Cependant, l'implantation de ce système en Chine est une affaire de longue haleine, étant donné la structure de l'administration chinoise : les provinces, municipalités et compagnies jouissent d'une certaine autonomie, et il faut donc traiter avec chacune d'elles séparément. Cela exige de la part de l'association française un gros travail de mailing et de prospection du marché.

Cette étude préindustrielle a également pour but de trouver des avantages supplémentaires qui pourraient tenter les utilisateurs chinois. Ainsi, Olympia a pu vendre cent terminaux à codage alphabétique grâce à l'imprimante à jet d'encre qui les équipait.

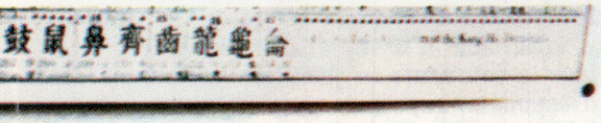
Lotus est un système relativement souple, dont la mémoire peut à tout moment être augmentée par l'entrée de nouveaux caractères, notamment dans des domaines spécialisés (pétrole, nucléaire, etc.). La graphie est mémorisée lors de sa première utilisation, et on lui affecte un numéro à deux octets.

Pour satisfaire leurs besoins, les Chinois exigent en général 6 000 caractères. Toutefois, des mémoires plus légères, et meilleur marché, pourraient suffire à certains utilisateurs.

Le champ d'application de Lotus est vaste. Au-delà d'un outil de communication, il peut servir de terminal de dialogue avec les bases de données en chinois, jouer un rôle dans l'enseignement assisté par ordinateur, en bureautique, voire dans la traduction automatique.

L'objet visé par l'ALFA est de commercialiser une première série de plusieurs centaines de machines, ce qui permettrait de proposer un terminal à un prix de 50 000 F environ.

Le périphérique mis au point par ALFA comporte une tablette sur laquelle sont écrits les caractères dont les formes sont comparées à celles des caractères mémorisés dans l'ordinateur. Un clavier numérique permet d'entrer les chiffres et les codes des caractères moins employés.



Terminal de saisie de caractères chinois ALFA.

Les mutinés du Bounty

Nous avons déjà vu que de nombreux événements modifient le cours du voyage. Mais l'équipage peut aussi se mutiner.

Plusieurs hasards peuvent affecter, négativement ou positivement, le cours de notre voyage. Ils sont choisis chaque semaine de façon aléatoire, mais ne surviendront plus par la suite. Il reste pourtant un événement qui peut se produire à tout moment de la partie, dès lors que certaines conditions sont réunies. Si la vie à bord devient intolérable, l'équipage se mutinera.

Huit facteurs différents peuvent contribuer à cela. Le navire peut accepter jusqu'à 16 hommes, mais en fait tout nombre supérieur à 12 représente un excédent; les marins sont entassés, et donc mécontents. Si vous n'avez pas embauché de cuisinier, ou s'il meurt durant la traversée, ils devront de plus s'occuper de la préparation des repas; la nourriture ne sera pas de la meilleure qualité, d'où mécontentement supplémentaire!

Apercevoir un albatros est un heureux présage, et l'équipage en sera heureux; mais s'il est tué, cela vous portera malheur, et la mutinerie deviendra plus probable. Autres facteurs : la nourriture distribuée par demi-rations; la chute des ressources financières, inférieures aux salaires à payer; une durée de voyage supérieure à huit semaines. Dans tous les cas, les matelots se mettront à maugréer, et cela ira en empirant à mesure que le temps passe et que la situation se dégrade.

Il faut d'abord voir si toutes les conditions sont réunies pour que la rébellion éclate. Pour cela, nous définirons une variable MF, qui représente le facteur de mutinerie. Au début d'une semaine, chaque condition est testée par appel d'un sous-programme « mutinerie », à partir de la boucle principale du programme. Si le résultat de chaque test est positif, une valeur supplémentaire est ajoutée à MF, et la procédure continue jusqu'à ce que la variable atteigne une valeur de 100 : la mutinerie aura lieu à ce moment. Pour donner aux choses une allure plus « réaliste », un facteur aléatoire, qui peut aller jusqu'à 30, est inclus dans chaque analyse hebdomadaire.

La ligne 879 nous fait passer au sous-programme qui commence ligne 7200, et procède à l'examen du facteur de mutinerie, MF, dont la valeur est d'abord fixée à 0. Si l'équipage est mis aux demi-rations à n'importe quel stade du voyage, la variable H$ se voit affecter la valeur 0. Distribuer la nourriture en quantités suffisantes est extrêmement important pour les marins; même si les restrictions ne s'appliquent qu'à un type de provisions, H$ gardera cette valeur pour

Module dix : mutinerie

Programme mutinerie

```
879 GOSUB7200
```

Supplément à la boucle Voyage principal

```
7200 REM MUTINY
7210 MF=0
7215 IFH$="o" THENMF=MF+30
7220 NC=0
7225 FORT=1TO16
7228 IFTS(T,1)=5ANDTS(T,2)<>0ANDTS(T,2)<>-999THENN
C=1:T=16
7230 NEXT
7235 'IFNC=0THENMF=MF+30
7240 IFA$="Y"THENMF=MF-20
7245 IFB$="Y"THENMF=MF+30
7250 IFCN>12THENMF=MF+30
7255 IFWT>MOTHENMF=MF+30
7260 IFWK>8THENMF=MF+(WK-8)*10)
7275 MF=MF+INT(RND(1)*30)
7280 IFMF<75 THENRETURN
7282 PRINTCHR$(147)
7284 IFMF>100THEN7300
7285 S$="LES CONDITIONS DE VIE*":GOSUB9100
7286 S$="SE DEGRADENT*":GOSUB9100
7287 S$="ET CERTAINS MATELOTS*":GOSUB9100
7288 S$="PARLENT DE MUTINERIE !*":GOSUB9100
7290 PRINT:GOSUB9200
7292 S$=K$:GOSUB9100
7294 GETI$=IFI$=""THEN7294
7299 RETURN
7300 PRINTCHR$(147)
7305 PRINT:GOSUB9200
7310 S$="L'EQUIPAGE S'EST MUTINE*":GOSUB9100
7312 S$="PARCE QUE:*":GOSUB9100
7313 X=0
7314 IFH$<>"o"THEN7320
7315 GOSUB9200:X=X+1:PRINTX:
7316 S$="PENDANT TROP LONGTEMPS*":GOSUB9100
7318 S$="ILS ONT VECU SUR DES DEMI-RATIONS*":GOSUB9100
7320 IFNC<>0THEN7325
7321 GOSUB9200:X=X+1:PRINTX:
7322 S$="IL N'Y A PAS DE CUISINIER*":GOSUB9100
7324 S$="ET LA NOURRITURE EST INFECTE !*":GOSUB9100
7325 IFB$<>"o"THEN7330
7326 GOSUB9200:X=X+1:PRINTX:
7327 S$="VOUS AVEZ TUE L'ALBATROS !*":GOSUB9100
7330 IFCN<13THEN7335
7331 GOSUB9200:X=X+1:PRINTX:
7332 S$="LE NAVIRE EST SURPEUPLE*":GOSUB9100
7335 IFMO>=WT THEN7340
7336 GOSUB9200:X=X+1:PRINTX:
7337 S$="IL N'Y A PLUS ASSEZ D'OR*":GOSUB9100
7338 S$="POUR PAYER LES HOMMES*":GOSUB9100
7340 IFWK<=8 THEN7350
7341 GOSUB9200:X=X+1:PRINTX:
7342 S$="ILS SONT EN MER*":GOSUB9100
7343 S$=" DEPUIS PLUS DE 8 SEMAINES*":GOSUB9100
7350 PRINT:GOSUB9200
7360 S$="L'EQUIPAGE S'EMPARE DU NAVIRE*":GOSUB9100
7362 GOSUB9200
7363 S$="ET DISPARAIT AU LOIN *":GOSUB9100
7370 GOSUB9200
7372 S$="VOUS LAISSANT SEUL*":GOSUB9100
7373 S$="DANS UN CANOT...":GOSUB9100
7374 S$="PUISSIEZ-VOUS SURVIVRE !"
7375 PRINT:GOSUB9200
7380 PRINT"      FIN DE LA PARTIE"
7382 END
```

Variantes de basic

Spectrum :
Procédez aux modifications suivantes :

```
7282 CLS
7294 LET I$=INKEY$:IF I$=""THEN GO TO 7294
7300 CLS
```

BBC Micro :
Procédez aux modifications suivantes :

```
7282 CLS
7294 I$=GET$
7300 CLS
```

le restant du voyage. Le sous-programme ajoute 30 à MF chaque fois que H$ est égal à 0. C'est la tâche de la ligne 7215, qui incrémente MF si nécessaire.

Après quoi le sous-programme vérifie s'il y a un cuisinier à bord. Pour cela il attribue à la variable NC la valeur 0, puis crée une boucle allant de 1 à 16, et parcourt le premier élément du tableau TS(,), consacré à la force et à la catégorie de chaque matelot. Il est à la recherche d'un 5, qui indique un cuisinier. Si la force de celui-ci n'est pas de 0 ou de -999, le cuisinier est vivant, et NC prend la valeur 1. Si par contre vous n'avez engagé aucun maître queux, ou s'il est mort en cours de route, NC reste à 0, et, dans ce cas, la ligne 7235 ajoute 30 à MF.

Si l'albatros a été aperçu au cours de la traversée, A$ devient 0, ligne 6055 du sous-programme « albatros ». Cela permet de soustraire, ligne 7240, 20 de MF. L'albatros est en effet un heureux présage. Si vous avez tué l'oiseau, la ligne 6162 donne à B$ la valeur 0 — c'est en effet un mauvais signe — et la ligne 7245 augmente le facteur de mutinerie de 30. La ligne 7250 cherche à savoir s'il y a plus de douze membres d'équipage; si oui, elle incrémente MF de 30, tenant compte ainsi du mécontentement des hommes placés sur un navire surpeuplé.

La somme totale due en salaires est représentée par WT, et l'argent conservé par le capitaine dans son coffre par M0. La ligne 7255 a pour fonction de comparer les deux valeurs. Si WT est plus élevé que M0, MF se voit augmenté de 30. Le facteur de mutinerie est incrémenté de 10 pour chaque semaine supplémentaire de voyage. La ligne 7260 vérifie si le périple a déjà atteint les huit semaines. Si c'est le cas, elle soustrait 8 du nombre de semaines (WK), multiplie le résultat par 10, et ajoute le tout à MF. Cette variable peut aussi s'enrichir d'un facteur aléatoire compris entre 0 et 29, et qui est déterminé ligne 7275.

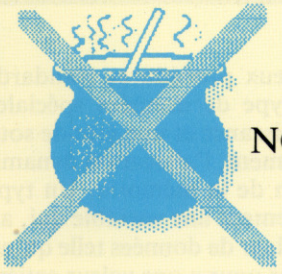
Toutes les conditions ayant été analysées, si MF est inférieur à 74, le programme repasse en boucle principale, ligne 7280. S'il est supérieur à 100, la ligne 7284 nous envoie à la ligne 7300, et la mutinerie éclate. Enfin si MF est compris entre 75 et 100, les lignes 7285 à 7288 vous informent que les conditions de vie se dégradent et que l'équipage, très mécontent, songe à se mutiner. On en revient ensuite en boucle principale.

Si la mutinerie a bel et bien lieu, le programme en détermine les causes, qu'il affiche à l'écran. La ligne 7314 voit si H$ est égal à 0, et vous informe que les hommes survivent sur des demi-rations depuis un certain temps. La ligne 7320 fait de même pour NC; si cette variable est égale à 0, c'est qu'il n'y a pas de cuisinier à bord, et la ligne 7324 précise que la nourriture était exécrable! La ligne 7325, de façon analogue, s'assure que l'albatros a été tué ou non; si oui, vous apprendrez que c'était un mauvais présage, qui a contribué à l'explosion de colère. Les lignes 7330 (vaisseau surpeuplé) et 7335 (plus assez d'argent pour payer les hommes) ont la même fonction, tout comme la ligne 7340 (le délai de huit semaines est dépassé).

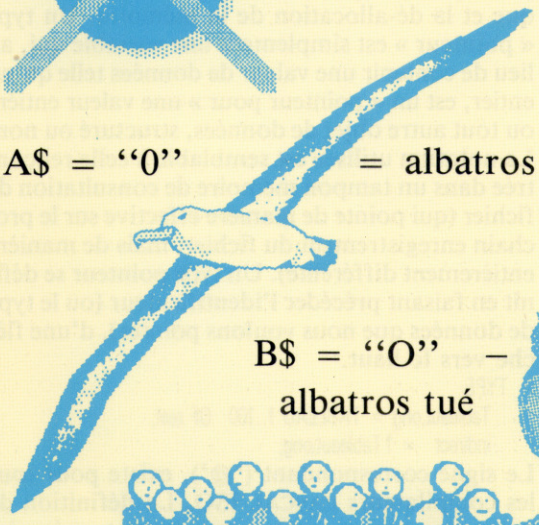
Les facteurs de la mutinerie

Huit facteurs contribuent à déterminer le moral de l'équipage, et permettent de définir un facteur de mutinerie, MF, pour chaque semaine du voyage. Si la combinaison de ces huit facteurs donne à MF une valeur supérieure à 100, la mutinerie éclate, et le capitaine du navire sera abandonné au gré des courants, à bord d'un petit canot de sauvetage.

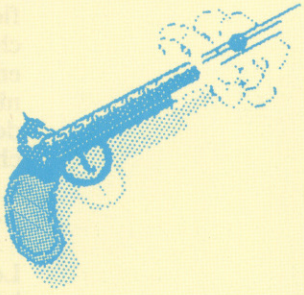
H$ = "O"
demi-rations



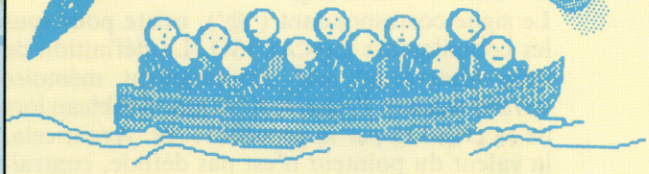
NC = 0 – pas de cuisinier –



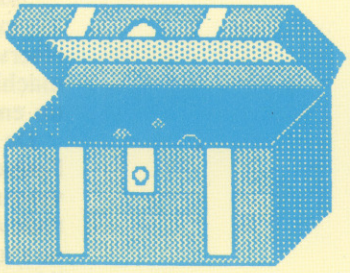
A$ = "0" = albatros repéré



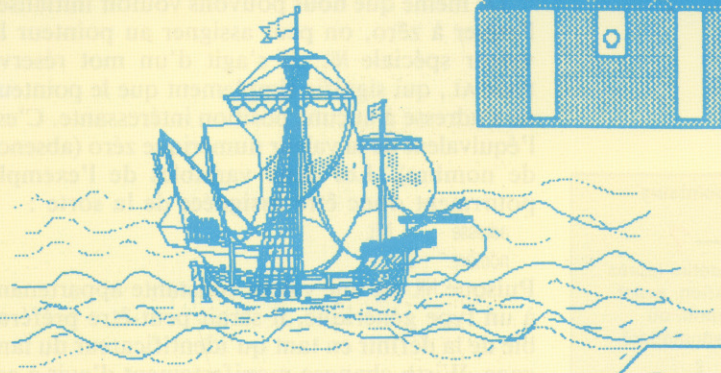
B$ = "O" – albatros tué

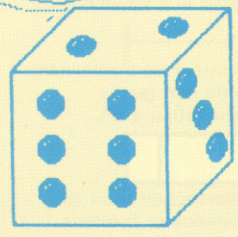


CN > 12 – vaisseau surpeuplé



WT > M0 – les salaires plus importants que la somme dans le coffre





WK > 8 – 1 voyage > 8 semaines

facteur aléatoire

Ian McKinnell

Allocation dynamique

Nous allons voir des concepts en PASCAL tels que la dé-allocation, les pointeurs, les sauts et les structures liées. Ces derniers nous permettent de traiter des fichiers de taille indéterminée.

Le PASCAL prévoit deux procédures standards (new et dispose) et un type de données spéciales offrant une grande puissance et une grande souplesse pour, respectivement, l'allocation dynamique et la dé-allocation de la mémoire. Un type « pointeur » est simplement une variable qui, au lieu de contenir une valeur de données telle qu'un entier, est un « pointeur pour » une valeur entière ou tout autre objet de données, structuré ou non. La notation utilisée est semblable à celle rencontrée dans un tampon mémoire de consultation de fichier (qui pointe de manière effective sur le prochain enregistrement du fichier, mais de manière entièrement différente). Un type pointeur se définit en faisant précéder l'identificateur (ou le type de données que nous voulons pointer), d'une flèche vers le haut.

```
TYPE
  TableauLong = TABLEAU 1..100  OF real;
  indirect    = ↑TableauLong;
```

Le signe correspondant ('@'), existe pour tous les compilateurs PASCAL ISO. La définition de type réserve un seul emplacement mémoire devant recevoir l'adresse d'un grand tableau lors de sa création par la procédure new. Avant cela, la valeur du pointeur n'est pas définie, contrairement à toute autre variable. Aussi :

```
VAR
  adresse : indirect;
  nombre : entier;
```

réserverait de la place pour une seule adresse machine (16 bits sur un micro à 8 bits), et pour un entier, les deux n'étant pas initialisés.

De même que nous pouvons vouloir initialiser l'entier à zéro, on peut assigner au pointeur la valeur spéciale NIL. Il s'agit d'un mot réservé PASCAL, qui signifie simplement que le pointeur ne s'adresse à aucune position intéressante. C'est l'équivalent de la valeur numérique zéro (absence de nombre). Les deux variables de l'exemple pourraient donc être assignées de la sorte :

```
adresse  := NIL
nombre   : 0;
```

Puisque NIL est une valeur constante appartenant à un type générique, il serait peut-être préférable de la définir en tant qu'identificateur du langage. Wirth changea manifestement d'avis à cet égard, le même mot étant dans MODULA-2 un identificateur prédéfini, et non un mot réservé. Pour illustrer l'utilisation de new et dispose, nous prendrons des entiers pour simplifier les choses.

Lorsqu'une variable pointeur est nécessaire pour désigner un nouvel élément, nous appelons la procédure PASCAL new qui lui alloue de la place mémoire et met son adresse dans le pointeur. L'élément de données est alors référencé en déréférençant le pointeur p par la notation p↑. Vous remarquez que la flèche vers le haut vient maintenant après l'identificateur du pointeur comme dans la notation concernant un tampon de fichier. Cette expression pourrait être énoncée de la sorte : « l'élément pointé par p ».

Ayant fini avec les données créées par les pointeurs initiateurs new, nous pouvons appeler la procédure PASCAL dispose. C'est l'inverse de new, dans la mesure où la mémoire est retournée à l'ensemble dynamique, la valeur du pointeur devenant indéfinie. Le programme DeuxPlusDeux le montre :

```
PROGRAMME DeuxPlusDeux (sorties);
TYPE
    pointeur   = ↑ entier;
VAR
    p1,
    p2         : pointeur;
    réponse    : entier;
BEGIN
  new (p1);
  p1 ↑ := 2;
  new (p2);
  p2 ↑ : = p1 ↑;
  réponse : = p1 ↑ + p2 ↑;
  WriteLn (p1 ↑, '+', p2 ↑, '=', réponse);
  dispose (p1);
  dispose (p2);
END
```

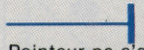
Cette méthode utilisée pour l'addition deux-plus-deux, met en évidence deux points très importants :

- Les variables dynamiques sont anonymes — il n'y a pas d'identificateur de variable (comme N), comportant la valeur 2, dans le programme. Ces éléments sont référencés indirectement.
- Après la deuxième procédure dispose, le seul espace mémoire utilisé est celui de l'entier (answer). Les données dynamiques n'existent plus.

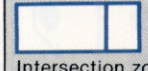
Si vous avez suivi les articles sur le langage assembleur, vous aurez compris que le mode indirect des pointeurs est analogue à l'adressage indirect niveau machine. Il existe cependant de grandes différences dans l'utilisation du mode indirect avec un langage aussi évolué que PASCAL. En premier lieu, nous ne savons pas (ou n'avons pas besoin de savoir) quelles seront les adresses effectives. La seule adresse absolue dont dispose le programmeur PASCAL est NIL. Nous sommes également libres d'utiliser, de rappeler — et, ultérieurement, de réutiliser — tout espace mémoire

Symboles pointeurs

Pointeur indéfini (après une déclaration, avant une affectation, ou après une annulation).



Pointeur ne s'adressant nulle part (après affectation à NIL).



Intersection zone de données et pointeur.

disponible sans avoir à nettoyer nous-mêmes la mémoire. La gestion de cette dernière est prise en charge par le PASCAL ; la seule information dont nous puissions avoir besoin étant la place mémoire disponible. On obtient cette information par la fonction non standard MemAvail, ou, sur certaines versions telles que le PASCAL ISO Acorn, par la fonction Free. Ces fonctions donnent le nombre d'octets encore disponibles en mémoire. Une autre fonction supplémentaire très utile est la fonction SizeOf (IdentificateurType), qui indique la taille en octets de tout type. La mémoire utilisateur est divisée en deux structures internes, le *tas* et la *pile*.

Le tas est utilisé pour les appels de procédures et de fonctions. Il stocke leur adresse de retour, les données locales et les valeurs restituées. Toutes les données dynamiques sont allouées sur la pile. Celle-ci fonctionne de manière similaire à celle du tas, à la différence près que le mode d'empilage n'est pas « dernier élément entré, premier sorti ». La pile commence à l'autre extrémité de la mémoire utilisateur. Elle s'accroît vers le tas. Une utilisation trop ambitieuse de la procédure new, et/ou de la récursion peut provoquer une collision pile/tas, mais on peut l'éviter de la sorte :

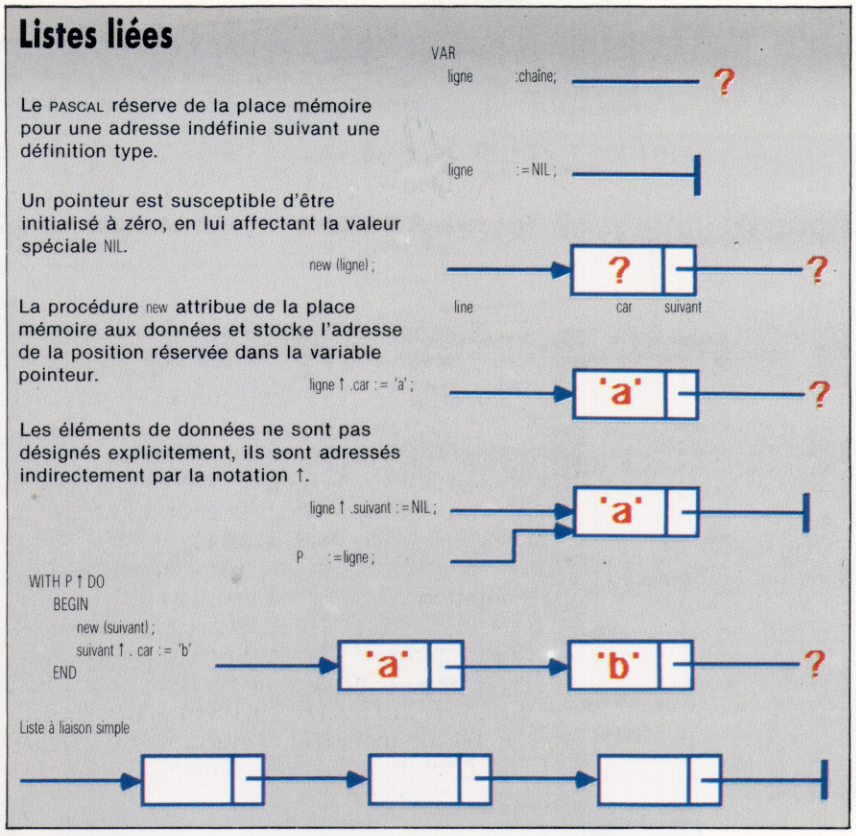
IF SizeOf (objet) > PerCent * MemAvail THEN... où objet est l'identificateur type des données à créer, et PerCent, une valeur autour de 0,7, allouant 30 % de la mémoire au tas, et 70 % à la pile. Si, comme c'est généralement le cas, MemAvail/Free fonctionne comme le niveau maximal pour la mémoire, il est possible de tenir en plus le compte de la mémoire récupérable grâce aux éléments supprimés en mémoire.

La puissance réelle des pointeurs est mise à l'œuvre lors de la création de structures liées telles les arborescences, les listes liées simplement ou doublement, les structures circulaires, etc. Si nous envisageons le problème des chaînes de caractères, nous pouvons, et nous le faisons souvent, utiliser un tableau de dimension fixe, par exemple de 80 caractères.

Un compilateur PASCAL doit distinguer des identificateurs de toutes longueurs. Ainsi, comment trouver un modèle précis pour cette caractéristique appartenant au monde réel, à savoir une taille variable ? La structure naturelle à utiliser serait un enregistrement de deux zones : une pour chaque élément de données (ici des caractères), et une autre zone pointeur de l'enregistrement suivant dans la liste, s'il existe.

```
TYPE
     chaîne    = ↑caractère;
     caractère = RECORD
                 car : car;
                 suivant :chaîne
                 END;
VAR
     ligne      : chaîne;
BEGIN
  ligne := NIL;
           etc.
```

Une chaîne vide est représentée simplement en assignant la valeur NIL à la chaîne. Toute autre



Liz Dixon

séquence de caractères supposera un nouvel enregistrement pour chaque caractère, et un autre pointeur sur l'enregistrement suivant. Le dernier enregistrement sera suivi d'une zone initialisée à la valeur NIL, ce qui permet de détecter la fin de la chaîne de caractères. Une procédure d'affichage de la chaîne pourrait être la suivante :

```
WHILE ligne <> NIL DO
  BEGIN
    write (ligne ↑. car);
    ligne := ligne ↑. suivante
  END
```

Vous remarquez que la structure des données est récursive, étant définie en termes d'elle-même. La zone pointeur ne peut recevoir un type qui aurait été pleinement défini, et une référence vers l'avant est donc autorisée.

Successivement
Le PASCAL offre au programmeur la possibilité de mettre en œuvre des « listes liées », structure des données très puissante dans laquelle chaque élément est pointé sur le suivant de la liste. Les listes peuvent être liées simplement (entre deux éléments successifs), liées doublement (c'est-à-dire que chaque élément dispose de pointeurs affectés à la fois vers le suivant et le précédent éléments de la liste), ou liées de façon cyclique.

Programme de liste cyclique

```
PROGRAM liste cyclique ( entrées, sorties );
{-------------------------------------------------
____ But : insertion d'enregistrements de
____ données de types divers, dans une liste
____ cyclique d'allocation dynamique. Les données
____ sont placées (IN) selon (OF) une clef-Zone
____ alphabétique (Nom) . Ainsi, le tri d'algorithmes
____ est superflu.
-------------------------------------------------}
CONST
      LongueurChaine =  25;
      Espace         = ' ';

TYPE
      Cardinal       = 0 .. EntierMaxI
      TailleChaine   = 1 .. LongueurChaine
      chaine         = TABLEAU LOT [ TableChaine ]
                       OF charI
      Objet          = RECORD
                           Nom     : Chaine
                           ( .. autres zones )
                           dette   : CardinaI
                       END; ( objet )
      pointeur       = ↑intersection;
```

Ligne circulaire
Ce programme permet d'insérer des enregistrements de données de types divers, dans une liste cyclique d'allocation dynamique. Les données sont placées selon une zone clé alphabétique (nom), ce qui rend superflu le tri des algorithmes.

```
        intersection          = RECORD
                                  élément     : objet;
                                  suivant     : pointeur
                                END; ( intersection )
  VAR   liste                 : pointeur ;

  (11111111111111111111111111111111111111)

  PROCEDURE  Lire-Objets   ( en-tête : pointeur );

  VAR
        données     : objet
        OK          : booléen

    (22222222222222222222222222222222)

  PROCEDURE   SauterBlancs  ( VAR  F  : texte);
     ( ignorer l'espace de tête)
  VAR
       fait     : boléen;
  BEGIN
    fait  := EoF ( F );

    WHILE NOT fait DO
      BEGIN
        fait  := ( F  = espace ) OR EoLn ( F );

        IF NOT fait THEN
          BEGIN
            get ( F );
            fait := EoF ( F )
          END
      END

  END;  ( sauterBlancs )
    (22222222222222222222222222222222)

PROCEDURE  LireCarte   ( VAR F  : texte
                         VAR N  : Cardinal;
                         VAR OK : booléen );
   ( prendre nombre non-négatif )
  TYPE
     unique    = 0 .. 9;
  VAR
     chiffres  : SET OF char;
     fait      : booléen;

     (33333333333333333333333)

   FUNCTION  Valeur ( chiffre   : char )      : unique
     ( convertir char en une valeur numérique )
     BEGIN
       Valeur  := ord ( chiffre ) - ord ( '0' )

     END;        ( valeur )
       (33333333333333333333333)

  FUNCTION  Lesal ( N         : Cardinal;
                    chiffre   : char )      : booléen;
       ( sommes nous prêts de la valeur max  TO)
        EntierMax
   BEGIN
     IF N = EntierMax DIV 10
       THEN
         Lesal := valeur ( chiffre )   = EntierMax MOD 10
       ELSE
         Lesal  := N   EntierMax DIV 10
   END;      ( Lesal )
      (333333333333222222222222)

BEGIN   ( LireCarte )

  chiffres :=k [ '0'  .. '9' ];
  OK := NOT EoF ( F );

  IF OK THEN ( trouvé un chiffre ? )
    OK := F ^ IN chiffres

  N := 0;
  fait := NOT OK;

  WHILE OK AND NOT fait DO
    BEGIN ( passer N en base 10 :)
      N := 10 * N + valeur ( F ^);
      get ( F );
      fait := EoF ( F );

      IF NOT fait THEN
        fait := NOT ( F ^ IN chiffres );

      IF NOT fait THEN
        OK := Lesal ( N, F ^ )
    END;

  IF NOT EoF ( F ) THEN
    ReadLn ( F )

END;        ( LireCarte )
   (22222222222222222222222222222222)

PROCEDURE  chaîne ( VAR S   : LireLigne );
   ( lire une ligne de caractères (DF) au clavier )
 VAR

     index   : 0 .. LongueurChaîne ;
     symbol  : char;
BEGIN
 SauterBlancs ( Entrée );
 index := 0;

 WHILE NOT EoLn AND (index < LongueurChaîne ) DO
   BEGIN ( mettre un caractère dans la chaîne )
     index  := succ ( index );
     read ( symbol );
     S [ index ] := symbol
   END;

 IF index < LongueurChaîne
   THEN ( remplir WITH NUL$ )
     FOR index := index + 1 TO LongueurChaîne DO
       S [ index ] := chr   0
   ELSE
     IF NOT EoLn THEN ( trop de caractères );
       WriteLn ( 'WARNING - chaînes d'entrée tronquée' );
 ReadLn

END;  ( LireLigne )
  (22222222222222222222222222222222)

PROCEDURE  Insert  ( liste    : pointeur
                     données  : objet
 VAR
      éclaireur
      liaison   : pointeur ;
      alpha     : chaîne
BEGIN ( mettre les données en mode en-tête : )
 liste ^ .éléments := données
 éclaireur := liste
 alpha     := éclaireur ^ . suivant  .  élément  .  Nom;

 WHILE données  .  Nom < alpha DO
   BEGIN ( passer en revue la liste )
     éclaireur := éclaireur ^ .  suivant
     alpha     := éclaireur ^ .  suivant ^ . (élément. Nom)
   END;
      ( pas de contrôle des doubles (FOR)
new   liaison );          ( créer un autre mode )
liaison ^ . élément := données ( insertion données )
liaison ^ . suivant := éclaireur  (dans la liste)

 END; ( Insertion )
   (22222222222222222211111111111111)

BEGIN  ( LireObjets )

 Ecrire ( ' Nom ? ' );
 LireLigne ( données . Nom );
       ( Arrêt sur une entrée nulle :)
WHILE données . Nom [ 1 ] <> chr ( 0 ) DO
  BEGIN
    REPEAT
      REPEAT
        write ( ' Dette ? ' : 20 ) ;

        IF EoLn ( entrée ) THEN
          ReadLn ( entrée );

        SauterBlancs ( entrée )

    UNTIL NOT EoLn ( entrée );

    LireCarte ( entrée , données . dette OK );

    IF NOT OK THEN ( pas de chiffres )
      WriteLn ( ' *** ERREUR  : 20,
            veuillez réécrire ( *** )
  UNTIL OK;

    Insertion ( en-tête données );
    WriteLn  ( '(RETURN Lorsque fait )' : 40 );


    write ( ' Nom ? ');
    ReadLine ( données . Nom )
  END

END;   ( LireObjets )
(11111111111111111111111111111111111111)

PROCEDURE (Affichage ( Liste : Pointeur );

 VAR

      Imprimer  : pointeur
BEGIN ( sauter d'abord l'intersection en-tête fictif : )
 print := liste ^ . suivant;
 page ( sortie );
 WriteLn ( 'liste ordonnée :' );
 WriteLn;
      ( autres données à imprimer )
 WHILE NOT (imprimer = liste ) DO
   BEGIN
     WITH imprimer ^ . liste DO
       WriteLn ( dette : 8, Nom : 30 );

     WriteLn;
     ( imprimer := imprimer ^ . suivant )
   END

END;   ( afficher )
(11111111111111111111111111111111111111)

BEGIN    ( ListeCyclique - ProgrammePrincipal )

 new ( liste );  (création d'une intersection d'en-tête
                  fictive )
liste ^ . suivant := liste ; (instant sur elle-même )
page ( sortie );
WriteLn ( 'Enter donner :  nom d'abord );
WriteLn ( 'la dette est de ... :' );
WriteLn;
LireObjets (liste);  ( mettre données dans liste )
Display ( Liste );   ( l'imprimer dans l'ordre )
WriteLn ( '--- fait ---' : 40 )

END .
```

Beat generation

La puce dédiée au son du Commodore 64 ou puce SID est virtuellement un synthétiseur de son. Avec un logiciel sophistiqué, elle peut même produire de la parole comme un robot.

Le son du Commodore 64 est normalement transmis, *via* le plot RF, directement à la télévision. Cependant, la sortie son peut aussi être connectée, *via* le plot audio/vidéo, à un système hi-fi pour une lecture ou un enregistrement de bonne qualité. Nous allons voir les principes sous-jacents au logiciel permettant au C64 de se transformer en tambour.

Outre la création de sons complexes sous contrôle logiciel, la puce SID peut aussi accepter des entrées d'un signal audio externe. Ces signaux peuvent être générés par un matériel électronique externe (pouvant contenir d'autres puces SID) ou des instruments de musique tels que des guitares électriques. Ce signal externe peut être mixé à la sortie audio de la puce SID et traité à travers ses filtres. Nous recommandons toutefois de faire attention en utilisant les interfaces E/S, car une erreur de connexion des lignes externes pourrait gravement endommager l'ordinateur. Nous vous suggérons de consulter le manuel du fabricant avant de tenter toute connexion externe.

Pour commencer, voyons comment le son musical (périodique) peut être obtenu à partir de sons purs séparés. Considérons une autre méthode pour atteindre le même résultat. En contrôlant l'enveloppe d'une note périodique donnée, nous pouvons introduire les harmoniques selon différentes proportions. Effectivement, cela signifie que nous pouvons créer presque tous les sons périodiques que nous voulons, en modifiant quelques pics de l'enveloppe d'un oscillateur. Ces pics — ou contrôles — sont aussi connus sous le nom de « ADSR » (Attaque, Déclin, Soutien et Relâche).

Malheureusement, le BASIC résidant dans le C64 n'est pas bien équipé pour manier le son. La programmation du son sur l'ordinateur est effectuée par l'utilisation extensive des commandes PEEK et POKE. Si nous mettons la variable SID égale à 54272 (l'adresse de base de la puce SID), alors les adresses SID à SID+8 contrôleront la puce sonore — et donc tous les sons sur le Commodore 64.

Enfin, nous donnons un programme en langage machine pour créer un tambour à trois voix, commandé par interruptions. Ce code « coin » peut être contrôlé à partir du BASIC, mais continuera à faire jouer les tambours indépendamment du programme BASIC. Cela implique que, avec de légères modifications, il puisse servir à fournir des effets sonores de fond pour tout programme BASIC.

Sons musicaux

Le son atteint nos oreilles sous la forme de vibrations périodiques de la pression de l'air. Le nombre de vibrations par seconde s'appelle la « hauteur » — ou « fréquence » — du son. La limite inférieure audible par l'homme est d'environ 15 cycles par seconde (ou 15 hertz). Un son pur dont la fréquence est de 100 Hz paraît grave à l'oreille. Le *la* au-dessus du *do* médian est normalisé à 440 Hz. Si nous doublons la fréquence d'un son, nous élevons sa hauteur d'une octave. L'ouïe humaine embrasse près de 10,5 octaves. Les oscillateurs à trois tons de la puce SID embrassent un domaine de huit octaves environ — soit de 0 à 4 000 Hz.

Le physicien français, Jean Fourier (1768-1830), fut le premier à remarquer que toute onde périodique pouvait être considérée comme composée d'un son fondamental pur, auquel se mêlent des sons dont les fréquences sont des multiples de ce son fondamental. Ces sons s'appellent « harmoniques ». Ce qui donne à un son ou à une note son timbre caractéristique, ce sont les proportions dans lesquelles apparaissent les diverses harmoniques.

Une onde sinusoïdale pure, qui correspondrait à un son pur, est intrinsèquement un signal analogique, ce qui signifie qu'il n'est pas facile à produire sur un appareil numérique avec des niveaux de tensions de 0 ou 5 V. Donc, au lieu de générer des fréquences pures, puis de les mixer pour obtenir le son périodique souhaité, l'approche adoptée sur les micros familiaux est différente.

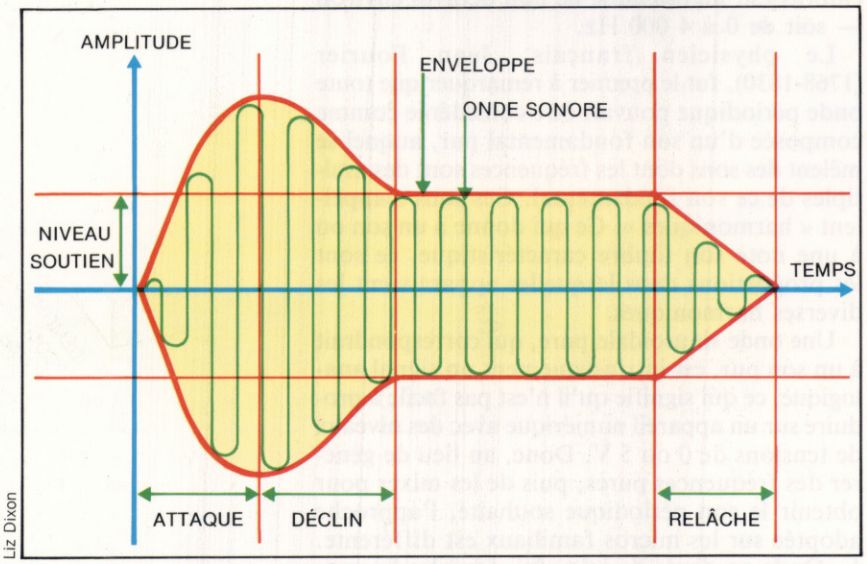
1. *Onde en dents de scie :* ici, tous les harmoniques sont présents. Le N^e harmonique a une intensité proportionnelle à $1/N$.
2. *Onde triangulaire :* elle ne contient que des harmoniques impairs. Pour N impair, le N^e harmonique a une intensité proportionnelle à $1/N^2$.
3. *Onde rectangulaire :* une telle onde a des harmoniques impairs proportionnels à $1/N$. En changeant la largeur d'impulsion, nous pouvons obtenir une grande diversité d'ondes rectangulaires, dont chacune a sa composition particulière.
4. *Bruit blanc :* un mélange aléatoire de fréquences, essentiellement utilisées pour des effets spéciaux.

Le contenu harmonique d'un son peut aussi être modifié par filtrage. La puce SID offre trois sortes de filtres : passe-bas, passe-bande et passe-haut. Par exemple, un filtre passe-bas passera toutes les fréquences en dessous d'une valeur prédéterminée et atténuera toutes celles qui sont au-

dessus de cette valeur. En utilisant ces possibilités et les contrôles d'enveloppe ADSR, presque tous les sons peuvent être produits.

Il y a diverses façons de construire des programmes générant de la parole. L'une d'elles, offrant une qualité acceptable et un vocabulaire illimité, est décrite ici. Dans cette méthode, les blocs constitutifs du langage — phonèmes — sont codés dans un tableau de valeurs ADSR. Il y a environ 37 phonèmes différents en français (ce nombre varie avec la langue : il y en a 52 en anglais), et leur codage n'est donc pas une tâche « phénoménale ». Puis un programme en langage machine est utilisé pour traduire un texte ASC (équivalent d'ASCII pour Commodore) en un flux de codes phonèmes, qui peut alors être envoyé à la puce SID à l'aide du tableau ADSR. Ce n'est pas si facile, d'abord parce que les règles qui sont utilisées pour traduire des textes en phonèmes sont très compliquées. La qualité de la parole obtenue dépendra grandement de la valeur de cette partie du programme. C'est un projet parfaitement réalisable pour le C64, et de nombreux produits utilisent cette technique.

Contrôle d'enveloppe



Liz Dixon

Enveloppes adressées
La qualité d'une note — ce qui permet de différencier entre, disons, un piano et un violon — dépend de la forme de l'enveloppe. Dans la synthèse électronique des sons, l'enveloppe est constituée de quatre étapes distinctes : attaque, déclin, soutien et relâche, ou ADSR. La longueur de chaque étape de l'enveloppe ADSR peut être définie en POKEant des valeurs dans les registres de la puce SID, ce qui nous permet de synthétiser les sons de différents instruments sur le C64.

Le diagramme ADSR donné ici montre la forme générale d'une note de musique, en insistant sur les caractéristiques que la puce SID nous permet de contrôler. Les quatre facteurs ADSR sont :

1. *Attaque :* le temps de montée d'une note.
2. *Déclin :* le temps pris par une note pour descendre à un niveau stable.
3. *Soutien :* le volume d'un niveau stable.
4. *Relâche :* le temps pris par le volume d'une note pour décroître du niveau soutien au niveau zéro.

Attaque, déclin et relâche d'une note sont chacun contrôlés par des quartets (4 bits) dans les registres de la puce SID. Cela implique que chacun de ces paramètres prend une valeur comprise entre 0 et 15. La relation entre les valeurs qui doivent être POKEées dans les registres SID et les synchronisations réelles sont données dans le tableau suivant :

Valeur	(temps/cycle) d'attaque Vitesse	(temps/cycle) déclin/relâche Vitesse de
0	2 ms	6 ms
1	8 ms	24 ms
2	16 ms	48 ms
3	24 ms	72 ms
4	38 ms	114 ms
5	56 ms	168 ms
6	68 ms	204 ms
7	80 ms	240 ms
8	100 ms	300 ms
9	250 ms	750 ms
10	500 ms	1.5 s
11	800 ms	2.4 s
12	1 s	3 s
13	3 s	9 s
14	5 s	15 s
15	8 s	24 s

La longueur de la section soutien est déterminée en utilisant une boucle retard. Les valeurs pour une note de violon pourraient être :

ADSR	Temps	POKE valeur
A	500 ms	10
D	300 ms	8
S	—	—
R	750 ms	9

Pour produire un son sur le C64 nous devons parcourir un minimum de six étapes. Par exemple :

• Étape 1 : mettre le volume à l'aide de :

```
POKE SID+24,15
```

• Étape 2 : sélectionner ADSR. Par exemple :

```
POKE SID+5,9:REM ATTAQUE/DÉCLIN VOIX#1
POKE SID+6,0:REM SOUTIEN/RELÂCHE VOIX#1
```

• Étape 3 : sélectionner la fréquence pour chaque oscillateur. Par exemple :

```
POKE SID+1,25:REM OCTET-HI FRÉQ VOIX#1
POKE SID,0    :REM OCTET-LO FRÉQ VOIX#1
```

• Étape 4 : sélectionner la forme d'onde désirée. Par exemple :

```
POKE SID+4,33:REM SÉLECTIONNER DENTS DE SCIE POUR
VOIX#1
```

A ce point, le son commence à jouer — ce qui se traduit par ouvrir la « porte ».

• Étape 5 : une boucle de retard pendant que la note joue au niveau soutien.

• Étape 6 : la forme d'onde doit être relâchée. Par exemple :

```
POKE SID+4,32:REM RELÂCHE DENTS DE SCIE
```

La façon la plus simple de programmer le Commodore 64 pour jouer un air est d'initialiser les valeurs de ADSR, et construire une boucle FOR qui lise (READ) les octets hi/lo de fréquence à partir des instructions DATA. En insérant des zéros dans les DATA, on peut varier le tempo pour différentes voix tout en maintenant le retard constant.

Machine à tambour

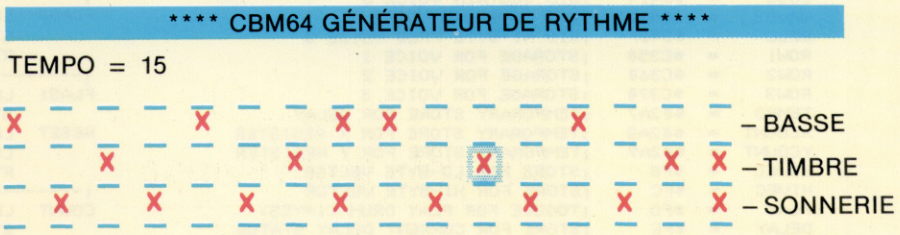
Le programme Machine à tambour que nous donnons ici utilise un « coin » IRQ, de sorte qu'une fois que nous avons mis les tambours en marche ils continuent à jouer, indépendamment de ce que nous faisons en BASIC. Le programme a aussi une manière intéressante de vérifier le code machine pour voir quelle touche a été appuyée — il inspecte le contenu de l'emplacement 197 ($00C5). C'est un emplacement en page zéro qui contient une valeur décrivant la dernière touche appuyée. Un programme en langage machine peut vous épargner le souci d'un appel noyau pour obtenir un appui de touche, pourvu que $00C5 soit échantillonné suffisamment souvent.

Notez que, puisque l'amplificateur BASIC ne vide pas le tampon de clavier à l'aide de GET ou INPUT, le programme doit mettre le pointeur à l'emplacement 198 ($00C6) à zéro pour la sortie. C'est l'emplacement de page zéro.

Lorsqu'on utilise le programme, à chaque tambour est attribué 16 cases de temps, lesquelles sont affichées sur une grille à l'écran. Le choix de 16 facilite la mise en œuvre d'un rythme de rock 4-4. Le curseur de lutin est déplacé dans l'une des cellules de la grille, et lorsqu'il est en position correcte, une case de temps peut être activée/désactivée avec la touche « Return », et le rythme mis à l'aide de la touche f1. Le menu à l'écran affiche les options disponibles. Étant donné que les sons sont commandés par interruption, le modèle de rythme peut être édité à l'écran tandis que le rythme joue.

Le programme BASIC inclut tout le code machine nécessaire au programme sous forme de DATA, et peut être tapé et exécuté tel quel. Nous fournissons aussi un listage de code source pour la partie en langage machine du programme. Si vous voulez taper et assembler le code source, alors les lignes 1520 à 1540 et les instructions DATA entre les lignes 1620 et 1940 peuvent être omises.

Pendant l'exécution du programme, essayez le modèle ci-dessous pour un rythme classique :



Programme générateur de rythme BASIC

```
1000 REM  ** GENERATEUR DE RYTHME **
1010 PRINTCHR$(147):REM CLEAR SCREEN
1020 GOSUB1510:REM SETUP M/C+SPRITE
1030 GOSUB1270:REM SETUP SCREEN
1040 SYS(49152):REM *INSERT WEDGE*
1050 REM SYS(49175) TO REMOVE WEDGE
1060 REM *MAIN LOOP*
1070 P=PEEK(197):REM CURRENT KEYPRESS
1080 IFP=37THENY=Y-16:SY=SY-1:IFY<95THENY=95:SY=0
1090 IFP=36THENY=Y+16:SY=SY+1:IFY>127THENY=127:SY=
2
1100 IFP=47THENX=X-16:SX=SX-1:IFX<28THENX=28:SX=0
1110 IFP=44THENX=X+16:SX=SX+1:IFX>268THENX=268:SX=
15
1120 Z=0:IFP=1THENZ=1
1130 PRINTCHR$(19);S$;"TEMPO =";PEEK(679);CHR$(157
);"   "
1140 POKEVIC+16,X/256:POKEVIC,XAND255
1150 POKEVIC+1,Y
1160 REM**COMPUTE SCREEN LOC FROM SPRITE X,Y POS
1170 SC=B+INT((Y-50)/8+1)*40+INT((X-18)/8)
1180 IFSY=0THENROW=50000:REM BASS DRUM
1190 IFSY=1THENROW=50016:REM SNARE DRUM
1200 IFSY=2THENROW=50032:REM COW BELL
1210 IFZ=1ANDPEEK(SC)=32THENPOKESC,24:POKEROW+SX,1
:GOTO1230
1220 IFZ=1ANDPEEK(SC)=24THENPOKESC,32:POKEROW+SX,0
1230 IFP=51THENGOSUB1270:FORI=0TO50:POKE50000+I,0:
NEXT
1240 IFP=57THENPOKE198,0:POKEVIC+21,0:PRINTCHR$(14
7):END
1250 GOTO1070
1260 REM *SET UP SCREEN*
1270 PRINTCHR$(19);
1280 S$=CHR$(17)+CHR$(17)+CHR$(17)
1290 PRINTTAB(3)"
"
1300 PRINTCHR$(18)TAB(3)" **** CBM64  RHYTHM GENER
ATOR **** "
1310 PRINT:PRINT:PRINT
1320 PRINT" - - - - - - - - - - - - - - - - "
1330 PRINT"                                -BASS"
1340 PRINT" - - - - - - - - - - - - - - - - "
1350 PRINT"                                -SNARE
"
1360 PRINT" - - - - - - - - - - - - - - - - "
1370 PRINT"                                -BELL"
1380 PRINT" - - - - - - - - - - - - - - - - "
1390 PRINT:PRINTTAB(8)"OPTIONS...."
1400 PRINTTAB(8)"[F1]    PLAY RHYTHM"
1410 PRINTTAB(8)"[F3]    DECREASE TEMPO"
1420 PRINTTAB(8)"[F5]    INCREASE TEMPO"
1430 PRINTTAB(8)"[F7]    STOP RHYTHM"
1440 PRINTTAB(8)"[CLR]   CLEAR CHART"
1450 PRINTTAB(8)"[ST]     END PROGRAM"
1460 PRINTTAB(8)"[<]     MOVE CURSOR LEFT"
1470 PRINTTAB(8)"[>]     MOVE CURSOR RIGHT"
1480 PRINTTAB(8)"[K]     MOVE CURSOR UP"
1490 PRINTTAB(8)"[M]     MOVE CURSOR DOWN"
1500 RETURN
1510 REM *SET UP M/CODE & SPRITE*
1520 FORI=49152TO49413
1530 READJ:C=C+J:POKEI,J:NEXTI
1540 READJ:IFC<>JTHENPRINT"DATA ERROR":END
1550 FORI=0TO62:READJ:POKE832+I,J:NEXTI
1560 VIC=53248:X=28:Y=95:B=1024:SID=54272
1570 FORI=0TO24:POKESID+I,0:NEXTI
1580 POKESID+24,15:POKEVIC+21,1:POKE2040,13
1590 POKE254,15:POKE679,15:POKEVIC+39,1
1600 FORI=0TO50:POKE50000+I,0:NEXT
1610 RETURN
1620 REM***M/C DATA***
1630 DATA120,173,20,3,133,251,173,21,3
1640 DATA133,252,169,36,141,20,3,169
1650 DATA192,141,21,3,88,96,120,165,251
1660 DATA141,20,3,165,252,141,21,3,88
1670 DATA96,32,177,192,165,197,201,3
1680 DATA208,3,32,135,192,165,197,201,4
1690 DATA208,3,32,140,192,165,197,201,5
1700 DATA208,3,32,155,192,165,197,201,6
1710 DATA208,3,32,166,192,32,87,192,142
1720 DATA168,2,140,169,2,76,49,234,165
1730 DATA253,208,1,96,198,254,240,1,96
1740 DATA32,149,192,224,16,208,3,32,144
1750 DATA192,185,80,195,240,3,32,184
1760 DATA192,185,96,195,240,3,32,210
1770 DATA192,185,112,195,240,3,32,236
1780 DATA192,200,234,232,96,169,0,133
1790 DATA253,96,169,1,133,253,162,0,160
1800 DATA0,96,173,167,2,133,254,96,173
1810 DATA167,2,201,255,240,3,238,167,2
1820 DATA96,173,167,2,201,1,240,3,206
1830 DATA167,2,96,174,168,2,172,169,2
1840 DATA96,169,14,141,6,212,169,32,141
1850 DATA2,212,169,66,141,4,212,169,3
1860 DATA141,1,212,169,65,141,4,212,96
1870 DATA169,7,141,12,212,169,12,141,13
1880 DATA212,169,128,141,11,212,169,65
1890 DATA141,8,212,169,129,141,11,212
1900 DATA96,169,2,141,19,212,169,13,141
1910 DATA20,212,169,18,141,18,212,169
1920 DATA100,141,15,212,169,17,141,18
1930 DATA212,96
1940 DATA32038:REM*CHECKSUM*
1950 REM**SPRITE CURSOR DATA**
1960 DATA127,254,0,127,254,0,127,254,0
1970 DATA112,14,0,112,14,0,112,14,0,112
1980 DATA14,0,112,14,0,112,14,0,112,14
1990 DATA0,112,14,0,127,254,0,127,254,0
2000 DATA127,254,0,0,0,0,0,0,0,0,0,0,0
2010 DATA0,0,0,0,0,0,0,0,0,0,0
```

Listage d'assemblage

```
;++++++++++++++++++++++++++
;++++++++++++++++++++++++++
;++                      ++
;++  MACHINE TAMBOUR CBM  ++
;++      CODE SOURCE     ++
;++                      ++
;++++++++++++++++++++++++++
;++++++++++++++++++++++++++

;
VOL     =  $D418    ;SID VOLUME
ATT1    =  $D405    ;SID ATTACK VOICE 1
SUS1    =  $D406    ;SID SUSTAIN VOICE 1
PULSE   =  $D402    ;SID PULSE RATE VOICE 1
WAVE1   =  $D404    ;SID WAVEFORM VOICE 1
BASS    =  $D401    ;SID HI-BYTE FREQ VOICE 1
ATT2    =  $D40C    ;SID ATTACK VOICE 1
SUS2    =  $D40D    ;SID SUSTAIN VOICE 2
WAVE2   =  $D40B    ;SID WAVEFORM VOICE 2
SNARE   =  $D408    ;SID HI-BYTE FREQ VOICE 2
ATT3    =  $D413    ;SID ATTACK VOICE 3
SUS3    =  $D414    ;SID SUSTAIN VOICE 3
WAVE3   =  $D412    ;SID WAVEFORM VOICE 3
BELL    =  $D40F    ;SID HI-BYTE FREQ VOICE 3
ROW1    =  $C350    ;STORAGE FOR VOICE 1
ROW2    =  $C360    ;STORAGE FOR VOICE 2
ROW3    =  $C370    ;STORAGE FOR VOICE 3
TEMPO   =  $02A7    ;TEMPORARY STORE FOR DELAY
XCOUNT  =  $02A8    ;TEMPORARY STORE FOR X REGISTER
YCOUNT  =  $02A9    ;TEMPORARY STORE FOR Y REGISTER
LOVEC   =  $FB      ;STORE FOR LO-BYTE VECTOR
HIVEC   =  $FC      ;STORE FOR HI-BYTE VECTOR
PLAY    =  $FD      ;TOGGLE FOR PLAY DRUM (1=YES)
DELAY   =  $FE      ;STORE FOR CURRENT DELAY STATUS
KEY     =  $C5      ;CURRENT KEY PRESSED
;-------------------------------------------------
*       =  $C000    ;ASSEMBLE FROM 49152 (DECIMAL)
;-------------------------------------------------
;SET UP WEDGE
;-------------------------------------------------
        SEI         ;DISABLE INTERRUPT REQUEST
        LDA $0314   ;GET CONTENTS OF VECTOR LO-BYTE
        STA LOVEC   ;STORE IN LOVEC
        LDA $0315   ;GET CONTENTS OF VECTOR HI-BYTE
        STA HIVEC   ;STORE IN HIVEC
        LDA #<WEDGE ;GET LO-BYTE WEDGE START ADDR.
        STA $0314   ;STORE IN IRQ VECTOR LO-BYTE
        LDA #>WEDGE ;GET HI-BYTE OF WEDGE START
        STA $0315   ;STORE IN IRQ VECTOR HI-BYTE
        CLI         ;RESUME INTERRUPT REQUEST
        RTS         ;RETURN
;-------------------------------------------------
;REMOVE WEDGE
;-------------------------------------------------
        SEI         ;DISABLE INTERRUPT REQUEST
        LDA LOVEC   ;GET ORIGINAL VALUE FROM LOVEC
        STA $0314   ;STORE IN IRQ VECTOR LO-BYTE
        LDA HIVEC   ;GET ORIGINAL VALUE FROM HIVEC
        STA $0315   ;STORE IN IRQ VECTOR HI-BYTE
        CLI         ;RESUME INTERRUPT REQUEST
        RTS         ;RETURN
;-------------------------------------------------
;MAIN LOOP
;-------------------------------------------------
WEDGE   JSR REG     ;GOSUB REG
        LDA KEY     ;WHICH KEY PRESSED ?
        CMP #$03    ;IS IT FUNCTION KEY #1 ?
        BNE CONT1   ;BRANCH IF NOT TRUE
        JSR FLAG0   ;GOSUB FLAG0
CONT1   LDA KEY     ;WHICH KEY PRESSED ?
        CMP #$04    ;IS IT FUNCTION KEY #7 ?
        BNE CONT2   ;BRANCH IF NOT TRUE
        JSR FLAG1   ;GOSUB FLAG1
CONT2   LDA KEY     ;WHICH KEY PRESSED ?
        CMP #$05    ;IS IT FUNCTION KEY #3 ?
        BNE CONT3   ;BRANCH IF NOT TRUE
        JSR ADD     ;GOSUB ADD
CONT3   LDA KEY     ;WHICH KEY PRESSED ?
        CMP #$06    ;IS IT FUNCTION KEY #5 ?
        BNE CONT4   ;BRANCH IF NOT TRUE
        JSR MINUS   ;GOSUB MINUS
CONT4   JSR REST    ;GOSUB REST
        STX XCOUNT  ;STORE VALUE OF X REGISTER
        STY YCOUNT  ;STORE VALUE OF Y REGISTER
        JMP $EA31   ;JUMP BACK TO INTERRUPT
;-------------------------------------------------
;CHECK ROUTINE
;-------------------------------------------------
REST    LDA PLAY    ;GET TOGGLE VALUE
        BNE BEGIN   ;IF EQUAL TO 1 THEN BRANCH
        RTS         ;RETURN
;-------------------------------------------------
BEGIN   DEC DELAY   ;DECREMENT DELAY
        BEQ START   ;BRANCH IF ZERO
        RTS         ;RETURN
;-------------------------------------------------
START   JSR COUNT   ;GOSUB COUNT
        CPX #$10    ;END OF LOOP ?
        BNE CHECK   ;BRANCH IF NOT 0
        JSR RESET   ;GOSUB RESET
;-------------------------------------------------
CHECK   LDA ROW1,Y  ;GET VALUE OF ROW1 OFFSET BY Y
        BEQ NEXT1   ;BRANCH IF ZERO
        JSR DRUM1   ;GOSUB DRUM1
NEXT1   LDA ROW2,Y  ;GET VALUE OF ROW2 OFFSET BY Y
        BEQ NEXT2   ;BRANCH IF ZERO
        JSR DRUM2   ;GOSUB DRUM2
NEXT2   LDA ROW3,Y  ;GET VALUE OF ROW3 OFFSET BY Y
        BEQ NEXT3   ;BRANCH IF ZERO
        JSR DRUM3   ;GOSUB DRUM3
NEXT3   INY         ;INCREMENT OFFSET
        INX         ;INCREMENT LOOP COUNTER
        RTS         ;RETURN
;-------------------------------------------------
;SUBROUTINES
;-------------------------------------------------
FLAG0   LDA #$00    ;STORE ZERO -
        STA PLAY    ;IN PLAY
        RTS         ;RETURN
;-------------------------------------------------
FLAG1   LDA #$01    ;STORE 1 -
        STA PLAY    ;IN PLAY
RESET   LDX #$00    ;RESET X REGISTER
        LDY #$00    ;RESET Y REGISTER
        RTS         ;RETURN
;-------------------------------------------------
COUNT   LDA TEMPO   ;GET VALUE OF TEMPO
        STA DELAY   ;STORE IN DELAY
        RTS         ;RETURN
;-------------------------------------------------
ADD     LDA TEMPO   ;GET VALUE OF TEMPO
        CMP #$FF    ;COMPARE RESULT WITH 255
        BEQ CONT5   ;BRANCH IF TRUE
        INC TEMPO   ;INCREMENT TEMPO
CONT5   RTS         ;RETURN
;-------------------------------------------------
MINUS   LDA TEMPO   ;GET VALUE OF TEMPO
        CMP #$01    ;COMPARE RESULT WITH 1
        BEQ CONT6   ;BRANCH IF TRUE
        DEC TEMPO   ;DECREMENT TEMPO
CONT6   RTS         ;RETURN
;-------------------------------------------------
REG     LDX XCOUNT  ;STORE VALUE OF XCOUNT IN X REG
        LDY YCOUNT  ;STORE VALUE OF YCOUNT IN Y REG
        RTS         ;RETURN
;-------------------------------------------------
;DRUM SOUND ROUTINES
;-------------------------------------------------
DRUM1   LDA #$0E    ;SET
        STA SUS1    ;
        LDA #$20    ;UP
        STA PULSE   ;
        LDA #$42    ;BASS
        STA WAVE1   ;
        LDA #$03    ;DRUM
        STA BASS    ;
        LDA #$41    ;AND PLAY
        STA WAVE1   ;
        RTS         ;RETURN
;-------------------------------------------------
DRUM2   LDA #$07    ;SET
        STA ATT2    ;
        LDA #$0C    ;UP
        STA SUS2    ;
        LDA #$80    ;SNARE
        STA WAVE2   ;
        LDA #$41    ;DRUM
        STA SNARE   ;
        LDA #$81    ;AND PLAY
        STA WAVE2   ;
        RTS         ;RETURN
;-------------------------------------------------
DRUM3   LDA #$02    ;SET
        STA ATT3    ;
        LDA #$0D    ;UP
        STA SUS3    ;
        LDA #$12    ;COW
        STA WAVE3   ;
        LDA #$64    ;BELL
        STA BELL    ;
        LDA #$11    ;AND PLAY
        STA WAVE3   ;
        RTS         ;RETURN
        .END
```

Page manquante
(publicité)

Page manquante
(publicité)